

# DLV11J

TEST  
CVDLAB0

AH-E189B-MC

COPYRIGHT 78-79

FICHE 1 OF 1

JAN 1979

**digital**

MADE IN USA

|    |    |    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|----|----|-----|
| 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10  |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20  |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30  |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40  |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50  |
| 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60  |
| 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70  |
| 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80  |
| 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90  |
| 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 |

.REM %

IDENTIFICATION

PRODUCT CODE: AC-E188B-MC  
PRODUCT NAME: CVDLAB0 DLV11-J TEST  
PRODUCT DATE: NOV 1978  
MAINTAINER: DIAGNOSTIC ENGINEERING

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1978, 1979 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

|         |       |         |         |
|---------|-------|---------|---------|
| DIGITAL | PDP   | UNIBUS  | MASSBUS |
| DEC     | DECJS | DECTAPE |         |

TABLE OF CONTENTS

|     |                                     |
|-----|-------------------------------------|
| 1.0 | GENERAL PROGRAM INFORMATION.        |
| 1.1 | PROGRAM PURPOSE (ABSTRACT).         |
| 1.2 | SYSTEM REQUIREMENTS.                |
| 1.3 | RELATED DOCUMENTS AND STANDARDS.    |
| 1.4 | DIAGNOSTIC HIERARCHY PREREQUISITES. |
| 1.5 | ASSUMPTIONS.                        |
| 2.0 | OPERATING INSTRUCTIONS.             |
| 2.1 | LOADING AND STARTING PROCEDURES.    |
| 2.2 | SPECIAL ENVIRONMENTS.               |
| 2.3 | OPERATIONAL SWITCH SETTINGS         |
| 2.4 | PROGRAM OPTIONS.                    |
| 2.5 | EXECUTION TIMES.                    |
| 2.6 | POWER FAIL.                         |
| 3.0 | ERROR INFORMATION.                  |
| 3.1 | ERROR REPORTING PROCEDURE.          |
| 3.2 | ERROR HALTS.                        |
| 4.0 | PERFORMANCE AND PROGRESS REPORTS.   |
| 4.1 | PERFORMANCE REPORTS.                |
| 5.0 | DEVICE INFORMATION TABLES.          |
| 6.0 | SUMMARY OF TESTS.                   |

1.0 GENERAL PROGRAM INFORMATION.  
-----

1.1 PROGRAM PURPOSE (ABSTRACT).

THIS DIAGNOSTIC IS A LOGIC TEST TO VERIFY THE OPERATION OF THE  
DLV11-J SERIAL INTERFACE.  
TESTING IS DONE IN TWO DISTINCT PHASES:

1. ALL SELECTED CHANNELS PER DLV11-J MODULE ARE TESTED INDIVIDUALLY
2. THE DLV11-J MODULE IS TESTED AS A WHOLE FOR CHANNEL INTERACTION PROBLEMS. THIS DIAGNOSTIC IS DESIGNED TO TEST AND DETECT ERRORS TO THE LOGIC LEVEL (NOT TO THE CHIP LEVEL).

THIS DIAGNOSTIC OPERATES UP TO 2 DLV11-J SERIAL LINE INTERFACES  
CONFIGURED AT CONSECUTIVE BASE ADDRESSES.  
THE PROGRAM WILL DO AUTO-SIZING IF THE DEVICE MAP '\$DEVN' 0.  
ALTERNATELY, THE OPERATOR CAN SELECT OR DESELECT INDIVIDUAL MODULES  
& CHANNELS BY SETTING THE PROPER BITS IN '\$DEVN' (SEE PROGRAM OPTIONS SEC. 2.4)  
BEFORE RUNNING THE PROGRAM.  
BY THIS METHOD, THE PROGRAM WILL BYPASS AUTO-SIZING AND TEST ONLY  
THOSE MODULES/CHANNELS IT FINDS IN \$DEVN.

IN EITHER CASE, THE PROGRAM WILL PRINT OUT ALL MODULES & CHANNELS  
TO BE TESTED BEFORE PROCEEDING.

THE OPERATOR MUST INSTALL DATA WRAP AROUND CONNECTORS TO DO  
DATA TESTING. TO BYPASS DATA TESTS, THE OPERATOR MUST MODIFY  
'\$USWR' (USER SWITCH REGISTER), SEE PROGRAM OPTIONS SEC. 2.4)

THE DEFAULT ADDRESSES & VECTORS ARE AS FOLLOWS:

177560 -CONSOLE INTERFACE DEVICE ADDRESS  
176500 -FIRST SERIAL CHANNEL ADDRESS OF UP TO 8 CONSECUTIVE  
SERIAL LINE DEVICES.

60 - VECTOR FOR CONSOLE DEVICE INTERFACE.  
300 - VECTOR FOR FIRST OF 16 DEVICES.

THIS PROGRAM IS DESIGNED TO RUN ON ANY Q-BUS PDP-11 WITH 4K OF  
MEMORY AND A DLV11-J (Q-BUS) MODULE. IT CAN RUN UNDER XXDP &  
APT MONITORS, AND ON PROCESSORS WITH NO HARDWARE  
SWITCH REGISTER.

## 1.2 SYSTEM REQUIREMENTS.

### HARDWARE REQUIREMENTS:

ANY Q-BUS PDP-11 FAMILY PROCESSOR  
4K MEMORY - MINIMUM  
A SPECIAL DATA WRAP AROUND CONNECTOR OR EQUIVALENT  
(REQ'D IF DATA WRAP AROUND TESTS DESIRED)

IF CHANNEL 3 IS CONFIGURED AS THE CONSOLE:  
TESTS 6-12, 14-17, 21, 22 ARE BYPASSED.

IF DATA WRAP AROUND TESTS ARE BYPASSED:  
TESTS 7-12, 14-17, 21, 22 ARE BYPASSED.

IF CHANNEL 3 IS CONFIGURED AS THE CONSOLE & A VT-100 IS  
THE CONSOLE DEVICE, SWITCH THE VT-100 TO 'JUMP SCROLL' MODE  
TO AVOID LOOSING CHARACTERS ON THE SCREEN.

### SOFTWARE REQUIREMENTS:

THIS DIAGNOSTIC IS DESIGNED TO RUN IN ANY OF THE  
FOLLOWING WAYS:  
STAND ALONE  
WITH APT MONITOR  
WITH XXDP MONITOR (CHAINABLE IF RENAMED TO .BIC EXTENSION)

THIS DIAGNOSTIC IS NOT DESIGNED TO RUN WITH THE  
DIAGNOSTIC SUPERVISOR.

## 1.3 RELATED DOCUMENTS AND STANDARDS.

|                                                  |                |
|--------------------------------------------------|----------------|
| DIAGNOSTIC ENGINEERING STANDARDS AND CONVENTIONS | 175-003-009-02 |
| APT                                              | MD-11-DZZMA    |
| SYSMAC                                           | MD-11-DZQAC    |

## 1.4 DIAGNOSTIC HIERARCHY PREREQUISITES.

NO SPECIAL DIAGNOSTICS ARE REQUIRED TO RUN BEFORE THIS, BUT  
THE PROCESSOR, MEMORY, AND BUS ARE ASSUMED TO BE FULLY  
OPERATIONAL.

## 1.5 ASSUMPTIONS.

THIS DIAGNOSTIC ASSUMES THAT THE OPERATOR HAS INITIALIZED  
LOCATION '\$USWR' AND '\$DEVM' TO THE PROPER VALUES. (SEE SEC. 2.4)

## 2.0 OPERATING INSTRUCTIONS.

### 2.1 LOADING AND STARTING PROCEDURES.

USE STANDARD PROCEDURE FOR PDP-11 ABSOLUTE BINARY FORMATTED MEDIA.

THERE ARE 2 STARTING LOCATIONS FOR THIS DIAGNOSTIC:

1. LOCATION 200 FOR ALL NORMAL TESTING STARTS & RESTARTS.
2. LOCATION 400 FOR INTERRUPT VECTOR DEBUG... OFFLINE ONLY.

THE 400 START SHOULD BE USED ONLY TO DEBUG FAULTY VECTOR ADDRESSES PUT OUT BY THE DV11-J DURING INTERRUPTS.

STARTING AT 400 WILL CAUSE THE ENTIRE VECTOR AREA, EXCEPT THAT NEEDED BY SYSMAC, TO BE OVERWRITTEN & TO POINT TO A COMMON SERVICE ROUTINE TO ENABLE LOOPING ON THE PROBLEM.

AS SOON AS TESTING STARTS, THE OPERATOR CAN CHANGE THE SWITCH REGISTER ONLY BY A 'BREAK' & MANUALLY LOADING LOCATION 176 (SWREG) WITH THE DESIRED CONTENTS (SEE SEC. 2.3) THEN DOING A 'P' TO PROCEED.

IT IS IMPORTANT THAT WHEN GOING BACK TO NORMAL TESTING A FRESH LOAD OF THE PROGRAM BE PERFORMED TO RESET THE VECTOR AREA.

THE USER CAN SELECT A SPECIFIC TEST TO BE EXECUTED BY SETTING BIT 8 IN SWREG AND THE TEST NUMBER (IN OCTAL) IN BITS <7:0>.

(NOTE: ALL TESTS PREVIOUS TO THE SELECTED ONE ARE EXECUTED WITHOUT ITERATIONS.)

### 2.2 SPECIAL ENVIRONMENTS.

THIS DIAGNOSTIC FOLLOWS THE STANDARD PROCEDURE FOR RUNNING UNDER APT,XXDP MONITORS, AS DESCRIBED IN THEIR RESPECTIVE PROCEDURES MANUAL AND SYSMAC PACKAGE.

### 2.3 OPERATIONAL SWITCH SETTINGS

THE SOFTWARE SWITCH REGISTER (LOC. 176) IS USED FOR ALL OPERATIONAL SWITCH SETTINGS.

THIS CAN BE ACCOMPLISHED IN THE FOLLOWING WAY:

- 1) TYPE CONTROL G <^G>; THIS WILL ALLOW THE TTY TO ENTER DATA INTO LOC. 176 AT SELECTED POINTS WITHIN THE PROGRAM.
- 2) THE MACHINE WILL THEN TYPE: 'SWR=XXXXXX NEW=' (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER.)
- 3) AFTER THE 'NEW=' HAS BEEN TYPED THEN THE OPERATOR CAN DO ONE OF THE FOLLOWING AT THE TTY:
  - A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A <CR>. (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED). LEADING ZEROS NEED NOT BE TYPED, AND IF MORE THAN 6 DIGITS ARE TYPED THE LAST 6 WILL BE USED. IF A <CR> IS THE FIRST KEY DEPRESSED THE SOFTWARE SWITCH REGISTER CONTENTS WILL NOT BE CHANGED.
  - B) IF A CONTROL U <^U> IS DEPRESSED THEN THE PROGRAM WILL SEND YOU BACK TO STEP 3.
  - C) IF THE INPUT CHARACTER IS NOT ONE OF THE CHARACTERS MENTIONED ABOVE THEN A QUESTION MARK (?) WILL BE TYPED FOLLOWED BY A CARRAGE RETURN AND A LINE FEED SEQUENCE THEN PROCEED FROM STEP 3 (ERASING ALL PREVIOUS INPUT).
- 4) THE DIAGNOSTIC WILL CONTINUE ON TYPING <CR>.

NOTE: BECAUSE OF FREQUENT BUS RESETS IN THE PROGRAM, IT MAY BE NECESSARY TO DO 'CONTROL-G' SEVERAL TIMES. ALTERNATELY, A 'BREAK' & MANUALLY LOADING LOC. 176, FOLLOWED BY A 'P' TO PROCEED WILL ALSO WORK.

SOFTWARE SWITCH REGISTER OPTIONS (SWREG)

-----

|        |     |   |        |   |                                                          |
|--------|-----|---|--------|---|----------------------------------------------------------|
| BIT 15 | SET | - | 100000 | - | HALT ON ERROR                                            |
| 14     | SET | - | 40000  | - | LOOP ON TEST (TO BE USED ONLY WHILE TESTING IN PROGRESS) |
| 13     | SET | - | 20000  | - | INHIBIT ERROR TIMEOUTS                                   |
| 12     | SET | - | 10000  | - | ENABLE PERFORMANCE REPORTS                               |
| 11     | SET | - | 4000   | - | INHIBIT ITERATIONS                                       |
| 10     | SET | = | 2000   | - | BELL ON ERROR                                            |
| 9      | SET | - | 1000   | - | LOOP ON ERROR                                            |
| 8      | SET | - | 400    | - | LOOP ON TEST IN SWR<7:0>                                 |
| 7:0    |     |   |        |   | NUMBER OF TEST TO LOOP ON (USED WITH BIT 8)              |
|        |     |   |        |   | (ALL TESTS PREVIOUS TO THE SELECTED TEST                 |
|        |     |   |        |   | ARE EXECUTED FIRST WITH 1 ITERATION ONLY)                |

## 2.4 PROGRAM OPTIONS.

THIS PROGRAM REQUIRES THE ADDRESS OF THE FIRST RCSR (STORED AT '\$BASE') AND ITS INTERRUPT VECTOR (STORED AT '\$VECT1'). IT WILL BE ABLE TO ADDRESS ANY DLV11-J STARTING AT THE SPECIFIED BASE ADDRESS THRU 2 CONSECUTIVE MODULES.

EXAMPLES: \$BASE: 176500 (DEFAULT)  
 \$VECT1: 300 (DEFAULT)

LOCATION '\$DEVM' IS USED AS A BIT MAP TO INDICATE WHICH UNIT NUMBERS ARE PRESENT AND WILL BE TESTED.

\$DEVM (BITS 15 THRU 8 NOT USED)

|       |    |    |    |    |    |   |   | MODULE 2 |   |   |   | MODULE 1 |   |   |   |         |
|-------|----|----|----|----|----|---|---|----------|---|---|---|----------|---|---|---|---------|
| 15    | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7        | 6 | 5 | 4 | 3        | 2 | 1 | 0 | BIT     |
| ***** |    |    |    |    |    |   |   |          |   |   |   |          |   |   |   |         |
|       |    |    |    |    |    |   |   | 3        | 2 | 1 | 0 | 3        | 2 | 1 | 0 | CHANNEL |
| ***** |    |    |    |    |    |   |   |          |   |   |   |          |   |   |   |         |

A BIT MAP CAN BE ENTERED AT '\$DEVM' PRIOR TO STARTING THE PROGRAM. IF '\$DEVM' IS LEFT AT ITS DEFAULT VALUE (0), THE PROGRAM WILL DO AUTO-SIZING BEGINNING AT THE DEVICE ADDRESS IN '\$BASE' & WILL SETUP '\$DEVM' ACCORDINGLY. IN EITHER CASE, THE PROGRAM WILL TYPE OUT A LIST OF ALL THE MODULES/CHANNELS TO BE TESTED BEFORE PROCEEDING.

NOTE: SIZING IS PERFORMED ONLY ONCE AT THE BEGINNING OF THE PROGRAM. IF \$DEVM IS TO BE CHANGED, THE PROGRAM MUST BE RESTARTED AT 200 FOR IT TO BE EFFECTIVE. OTHERWISE, \$DEVM WILL BE IGNORED.

### OPTIONS

LOCATION \$USWR CONTAINS ALL THE USER SELECTABLE OPTIONS. THE VALUES IN THIS WORD MUST CONFORM TO THE ACTUAL BOARD CONFIGURATION.

| BITS   | DEFINITION                                       | \$USWR | DEFAULT VALUE                        |
|--------|--------------------------------------------------|--------|--------------------------------------|
| 0      | # OF DATA BITS TRANSMITTED<br>0 7 BITS, 1 8 BITS | 1      | = 8 BITS                             |
| 1      | PARITY ENABLED                                   | 0      | - NO                                 |
| 2      | EVEN ODD PARITY                                  | 0      | - ODD                                |
| 3      | BREAK DETECTION ENABLED                          | 1      | - YES                                |
| 4      | RUN DATA WRAP AROUND TESTS                       | 1      | - YES                                |
| 11:9   | CONSOLE DEVICE                                   | 1      | - YES - CONSOLE ON MODULE 1, CHAN. 3 |
| <11:9> | 0 CONSOLE NOT ON DLV11-J                         |        |                                      |
| <11:9> | 1 CONSOLE ON MODULE 1                            |        |                                      |
| <11:9> | 2 CONSOLE ON MODULE 2                            |        |                                      |

IMPORTANT:

1. FOR DIAGNOSTIC PURPOSES, ALL CHANNELS MUST BE CONFIGURED TO THE SAME BIT/WORD LENGTH.
2. THE 'SIZING' ROUTINE WILL SET THE APPROPRIATE CHAN 3 BIT IN \$DEVN IF BITS <11:9> ARE SET IN \$USWR.
3. IF RUNNING UNDER APT & THE CONSOLE IS ON THE DLV11-J, THE CONSOLE MUST NOT BE TESTED. THIS IS BECAUSE APT SENDS 'BREAKS' TO THE CONSOLE WHICH INTERFERES WITH THE TESTS.

FOR THIS CASE:

- A. ALLOW THE \$DEVN TO STAY EITHER AT ITS DEFAULT OF 0 OR CLEAR THE APPROPRIATE CHAN. 3 BIT.
- B. CLEAR BITS <11:9> IN \$USWR SO THE CONSOLE BIT WILL NOT BE SET INTO \$DEVN FOR TESTING.

# SUMMARY OF DEVICE ADDRESSES & VECTORS.

|                |           | DEV ADDRESS<br>(RCSR) | VECTORS    |           |
|----------------|-----------|-----------------------|------------|-----------|
|                |           |                       | REC        | XMIT      |
| MODULE #1      | CHANNEL 0 | \$BASE+00             | \$VECT1+00 | \$VECT1+4 |
|                | 1         | +10                   | +10        | +14       |
|                | 2         | +20                   | +20        | +24       |
|                | 3         | +30                   | +30        | +34       |
| MODULE #2      | CHANNEL 0 | +40                   | +40        | +44       |
|                | 1         | +50                   | +50        | +54       |
|                | 2         | +60                   | +60        | +64       |
|                | 3         | +70                   | +70        | +74       |
| CONSOLE DEVICE | 5         | \$TKS                 | TKVEC-60   | TPVEC-64  |

# SUMMARY OF USER LOCATIONS & DEFAULTS.

|         | LOC  | DEFAULT |                                         |
|---------|------|---------|-----------------------------------------|
| \$BASE  | 1250 | 176500  |                                         |
| \$VECT1 | 1244 | 300     |                                         |
| \$TKS   | 1144 | 177560  | FOR CONSOLE DEVICE ON MODULE UNDER TEST |
| TKVEC   |      | 60      | FOR CONSOLE DEVICE ON MODULE UNDER TEST |
| TPVEC   |      | 64      | FOR CONSOLE DEVICE ON MODULE UNDER TEST |
| \$USWR  | 1220 | 1031    | SEE SEC. 2.4                            |
| \$DEVM  | 1252 | 0       | SEE SEC. 2.4                            |
| SWREG   | 176  | 0       | SEE SEC. 2.3                            |

## 2.5 EXECUTION TIMES.

EXECUTION TIMES FOR AN LSI-11 PROCESSOR WITH ONE DLV11-J MODULE AT SHIPMENT CONFIGURATION:

CH. 0,1,2 AT 9600 BAUD.  
 CH. 3 (CONSOLE) AT 300 BAUD.

ARE: FIRST PASS- 30 SEC  
 ADDITIONAL PASSES 90 SEC

THE TEST TIME IS BAUD RATE DEPENDENT; HIGHER BAUD RATES RESULT IN SHORTER PASS TIMES.

## 2.6 POWER FAIL.

AUTO START FROM POWER FAIL IS NOT IMPLEMENTED IN THIS PROGRAM.

### 3.0 ERROR INFORMATION.

#### 3.1 ERROR REPORTING PROCEDURE.

SINCE THIS DIAGNOSTIC WAS DESIGNED TO FIT IN 4K OF MEMORY THE ERROR TYPEOUT IS VERY BRIEF. THE ORMAT OF THE ERROR TYPEOUT IS AS FOLLOWS:

TEST#\_\_\_\_,ERROR#\_\_\_\_,PC-\_\_\_\_,ADDRESS=\_\_\_\_,VECTOR-\_\_\_\_

WHERE ALL VALUES TYPED ARE OCTAL.  
THE ADDRESS AND VECTOR REFER TO THE FAILING CHANNEL.  
FOR FURTHER INFORMATION THE LISTING MUST BE CONSULTED.  
BITS 15,13,10 AND 9 OF THE SWITCH REGISTER (SWREG) CONTROL THE SEQUENCE OF EVENTS AFTER AN ERROR IS CAUGHT.

BIT 15 SET: CAUSES THE PROGRAM TO HALT IN THE ERROR ROUTINE.  
IF THE PROGRAM IS CONTINUED, IT WILL PROCEED FROM WHERE IT HALTED.

BIT 13 SET: DISABLES THE PRINTING OF THE ERROR MESSAGE.

BIT 10 SET: CAUSES THE BELL TO RING ON ERROR.

BIT 9 SET: CAUSES THE DIAGNOSTIC TO LOOP FROM BEGINNING OF TEST TO ERROR.

THE ERROR ROUTINE SUPPORTS THE CONTROL G <^G> FUNCTION.  
REFER TO SECTION 2.3 FOR DETAILS.

#### 3.2 ERROR HALTS.

THE ONLY HALT IN THIS DIAGNOSTIC IS IN THE ERROR ROUTINE, AND IS EXECUTED ONLY IF BIT 15 OF THE SWITCH REGISTER (SWREG) IS SET WHEN AN ERROR OCCURS.

4.0 PERFORMANCE AND PROGRESS REPORTS.  
-----

4.1 PERFORMANCE REPORTS. (BIT 12 SET IN THE SWITCH REGISTER 'SWREG')

AS EACH CHANNEL (4 CHANNELS/DLV11-J) COMPLETES ONE PASS OF  
THE DIAGNOSTIC THE FOLLOWING ITEMS ARE TYPED:

'CSR:----' : THE BASE ADDRESS OF THE LINE UNDER TEST  
'VECTOR:--' : THE ASSOCIATED VECTOR  
'ERRORS:--' : THE TOTAL NUMBER OF ERRORS ON THIS DEVICE  
ON THIS PASS.

AFTER ALL MODULES & CHANNELS TO BE TESTED HAVE BEEN EXERCISED,  
AN END PASS STATEMENT IS TYPED:

'END PASS#-----.'

EXAMPLE OF PRINTOUT ASSUMING: 2 DLV11-J MODULES  
ALL CHANNELS ENABLED  
SEPERATE CONSOLE DEVICE  
NO ERRORS

CVDLAB0 DLV11-J TEST

SWR- 000000 NEW- 10000<CR>

(ENABLE PERFORMANCE REPORTS)

WILL TEST:

MODULE 1 CHANNEL 0 1 2 3  
MODULE 2 CHANNEL 0 1 2 3

\*\* PHASE 1 SUMMARY \*\*

CSR: 176500, VECTOR: 000300, ERRORS: 0  
CSR: 176510, VECTOR: 000310, ERRORS: 0  
CSR: 176520, VECTOR: 000320, ERRORS: 0  
CSR: 176530, VECTOR: 000330, ERRORS: 0

\*\* PHASE 2 SUMMARY \*\*

CSR: 176500, VECTOR: 000300, ERRORS: 0

\*\* PHASE 1 SUMMARY \*\*

CSR: 176540, VECTOR: 000340, ERRORS: 0  
CSR: 176550, VECTOR: 000350, ERRORS: 0  
CSR: 176560, VECTOR: 000360, ERRORS: 0  
CSR: 176570, VECTOR: 000370, ERRORS: 0

\*\* PHASE 2 SUMMARY \*\*

CSR: 176540, VECTOR: 000340, ERRORS: 0

END PASS # 1

# 5.0 DEVICE REGISTERS.

|      | 15   | 14  | 13  | 12  | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|-----|-----|-----|----|----|---|---|---|---|---|---|---|---|---|---|
| RCSR |      |     |     |     |    |    |   |   |   |   |   |   |   |   |   |   |
|      |      |     |     |     |    |    |   |   |   |   |   |   |   |   |   |   |
| RBUF | ERRO | OR  | FR  | P   |    |    |   |   |   |   |   |   |   |   |   |   |
|      | R    | ERR | ERR | ERR |    |    |   |   |   |   |   |   |   |   |   |   |
| TCSR |      |     |     |     |    |    |   |   |   |   |   |   |   |   |   |   |
|      |      |     |     |     |    |    |   |   |   |   |   |   |   |   |   |   |
| TBUF |      |     |     |     |    |    |   |   |   |   |   |   |   |   |   |   |
|      |      |     |     |     |    |    |   |   |   |   |   |   |   |   |   |   |

## NOTES:

1. RCSR AT BASE ADDRESS (\$BASE)  
 RBUF AT \$BASE+2  
 TCSR AT \$BASE+4  
 TBUF AT \$BASE+6
2. BLANK BOXES INDICATE UNUSED AND RESERVED  
 BIT POSITIONS. SEE THE LISTING FOR AN  
 EXPLANATION OF THE BITS.
3. ORERR      OVERRUN ERROR  
 FRERR      FRAMING ERROR  
 PERR      PARITY ERROR

6.0 SUMMARY OF TESTS AND SPECIAL SUBROUTINES.  
-----

PHASE 1 TESTS

TEST 1 ADDRESSABILITY

-----

THIS TEST VERIFIES THAT ALL 4 REGISTERS OF THE CHANNEL  
UNDER TEST RESPOND TO THEIR ADDRESSES.

THE FOLLOWING 3 TESTS TEST ALL 'READ WRITE' BITS

TEST 2 BREAK - TCSR 0 SET, CLEAR, RESET

-----

TEST 3 XMITIE - TCSR 6 SET, CLEAR, RESET

-----

TEST 4 RCVRIE - RCSR 6 SET, CLEAR, RESET

-----

TEST 5 XMITRDY - TCSR 7 - IS SET BY INIT

-----

TEST 6 XMIT RDY - TCSR 7 - CLEARS WHEN TBUF IS LOADED  
WITH A CHARACTER AND THAT IT SETS WITHIN A  
REASONABLE AMOUNT OF TIME.

TEST 7 OUTPUTTING A CHAR FROM TBUF (WITH WRAP AROUND CONNECTED)  
RESULTS IN RCVRDONE SETTING WITHIN A  
REASONABLE AMOUNT OF TIME AND THAT RESET  
CLEARS THE BIT.

TEST 10 RCVRDONE IS CLEARED BY SETTING READER ENABLE

-----

TEST 11 RCVRDONE IS CLEARED BY READING RBUF

-----

TEST 12 OVERRUN & ERROR BIT - RBUF 14

-----

TEST 13 TRANSMITTER INTERRUPT LOGIC TEST

---- --

LOGICALLY THIS IS 4 SEPARATE TESTS

- A) DOES TRANSMITTER INTERRUPT LOGIC WORK
- B) AT PRIORITY OF 0
- C) AND ONLY ONCE
- D) BUT NOT WITH INTERRUPT ENABLE CLEAR

TEST 14 RECEIVER INTERRUPT LOGIC TEST THIS TEST COVERS ALL  
---- -- OF THE RECEIVER SIDE OF THE INTERRUPT LOGIC IN  
CHARACTER MODE.

TEST 15 TEST DATA WRAP AROUND: FLAG MODE.

---- --

TEST 16 TEST DATA WRAP AROUND: INTERRUPT MODE.

---- --

TEST 17 TEST BREAK DETECTION LOGIC TRANSMIT KNOWN CHAR

---- --

- A) TRANSMIT KNOWN CHAR WITH BREAK SET  
AND COMPARE RECEIVED WITH 0
- B) TEST FOR FRAMING ERROR ON BREAK
- C) IF PARITY IS ENABLED AND ODD PARITY IS SELECTED,  
CHECK TO BE SURE PARITY ERROR WAS GENERATED
- D) IF PARITY IS ENABLED AND EVEN PARITY IS SELECTED,  
CHECK TO BE SURE NO PARITY ERROR OCCURRED

TEST 20 NOT A TEST - SEND BACK TO LOOP

## PHASE 2 TESTS

TEST 21 TEST THAT CHANNELS INTERRUPT AT ASSIGNED PRIORITY

---- --

TEST 22 TEST DATA TRANSFERS WITH ALL ACTIVE LINES INTERRUPTING.

---- --

%

```

616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671

```

```

.TITLE CVDLAAO DLV11-J TEST
;*COPYRIGHT (C) 1978
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*PROGRAM BY GREGORY GLEZMAN & GARY PAPAIAW
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
;*

.SBTTL OPERATIONAL SWITCH SETTINGS
;*
;*      SWITCH      USE
;*      -----
;*      15          HALT ON ERROR
;*      14          LOOP ON TEST
;*      13          INHIBIT ERROR TYPEOUTS
;*      11          INHIBIT ITERATIONS
;*      10          BELL ON ERROR
;*      9           LOOP ON ERROR
;*      8           LOOP ON TEST IN SWR<7:0>
.SBTTL BASIC DEFINITIONS
;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
.EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL

;*MISCELLANEOUS DEFINITIONS
HT= 11                ;;CODE FOR HORIZONTAL TAB
LF= 12                ;;CODE FOR LINE FEED
CR= 15                ;;CODE FOR CARRIAGE RETURN
CRLF= 200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776            ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW
STKLMT= 177774         ;;STACK LIMIT REGISTER
PIRQ= 177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
DSWR= 177570          ;;HARDWARE SWITCH REGISTER
DDISP= 177570         ;;HARDWARE DISPLAY REGISTER

;*GENERAL PURPOSE REGISTER DEFINITIONS
R0= %0                ;;GENERAL REGISTER
R1= %1                ;;GENERAL REGISTER
R2= %2                ;;GENERAL REGISTER
R3= %3                ;;GENERAL REGISTER
R4= %4                ;;GENERAL REGISTER
R5= %5                ;;GENERAL REGISTER
R6= %6                ;;GENERAL REGISTER
R7= %7                ;;GENERAL REGISTER
SP= %6                ;;STACK POINTER
PC= %7                ;;PROGRAM COUNTER

;*PRIORITY LEVEL DEFINITIONS
PRO 0                 ;;PRIORITY LEVEL 0

```

|     |        |      |     |                    |
|-----|--------|------|-----|--------------------|
| 672 | 000040 | PR1= | 40  | ::PRIORITY LEVEL 1 |
| 673 | 000100 | PR2= | 100 | ::PRIORITY LEVEL 2 |
| 674 | 000140 | PR3= | 140 | ::PRIORITY LEVEL 3 |
| 675 | 000200 | PR4= | 200 | ::PRIORITY LEVEL 4 |
| 676 | 000240 | PR5= | 240 | ::PRIORITY LEVEL 5 |
| 677 | 000300 | PR6= | 300 | ::PRIORITY LEVEL 6 |
| 678 | 000340 | PR7= | 340 | ::PRIORITY LEVEL 7 |

679 ;\*'SWITCH REGISTER' SWITCH DEFINITIONS

|     |        |        |          |
|-----|--------|--------|----------|
| 680 |        | SW15=  | 100000   |
| 681 | 100000 | SW14=  | 40000    |
| 682 | 040000 | SW13=  | 20000    |
| 683 | 020000 | SW12=  | 10000    |
| 684 | 010000 | SW11=  | 4000     |
| 685 | 004000 | SW10=  | 2000     |
| 686 | 002000 | SW09=  | 1000     |
| 687 | 001000 | SW08=  | 400      |
| 688 | 000400 | SW07=  | 200      |
| 689 | 000200 | SW06=  | 100      |
| 690 | 000100 | SW05=  | 40       |
| 691 | 000040 | SW04=  | 20       |
| 692 | 000020 | SW03=  | 10       |
| 693 | 000010 | SW02=  | 4        |
| 694 | 000004 | SW01=  | 2        |
| 695 | 000002 | SW00=  | 1        |
| 696 | 000001 | .EQUIV | SW09,SW9 |
| 697 |        | .EQUIV | SW08,SW8 |
| 698 |        | .EQUIV | SW07,SW7 |
| 699 |        | .EQUIV | SW06,SW6 |
| 700 |        | .EQUIV | SW05,SW5 |
| 701 |        | .EQUIV | SW04,SW4 |
| 702 |        | .EQUIV | SW03,SW3 |
| 703 |        | .EQUIV | SW02,SW2 |
| 704 |        | .EQUIV | SW01,SW1 |
| 705 |        | .EQUIV | SW00,SW0 |

706 ;\*DATA BIT DEFINITIONS (BIT00 TO BIT15)

|     |        |        |            |
|-----|--------|--------|------------|
| 707 |        | BIT15= | 100000     |
| 708 | 100000 | BIT14= | 40000      |
| 709 | 040000 | BIT13= | 20000      |
| 710 | 020000 | BIT12= | 10000      |
| 711 | 010000 | BIT11= | 4000       |
| 712 | 004000 | BIT10= | 2000       |
| 713 | 002000 | BIT09= | 1000       |
| 714 | 001000 | BIT08= | 400        |
| 715 | 000400 | BIT07= | 200        |
| 716 | 000200 | BIT06= | 100        |
| 717 | 000100 | BIT05= | 40         |
| 718 | 000040 | BIT04= | 20         |
| 719 | 000020 | BIT03= | 10         |
| 720 | 000010 | BIT02= | 4          |
| 721 | 000004 | BIT01= | 2          |
| 722 | 000002 | BIT00= | 1          |
| 723 | 000001 | .EQUIV | BIT09,BIT9 |
| 724 |        | .EQUIV | BIT08,BIT8 |
| 725 |        | .EQUIV | BIT07,BIT7 |
| 726 |        |        |            |
| 727 |        |        |            |

```

728      .EQUIV BIT06,BIT6
729      .EQUIV BIT05,BIT5
730      .EQUIV BIT04,BIT4
731      .EQUIV BIT03,BIT3
732      .EQUIV BIT02,BIT2
733      .EQUIV BIT01,BIT1
734      .EQUIV BIT00,BIT0
735
736      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
737      ERRVEC= 4          ;;TIME OUT AND OTHER ERRORS
738      RESVEC= 10         ;;RESERVED AND ILLEGAL INSTRUCTIONS
739      TBITVEC=14         ;;'T' BIT
740      TRTVEC= 14         ;;TRACE TRAP
741      BPTVEC= 14         ;;BREAKPOINT TRAP (BPT)
742      IOTVEC= 20         ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
743      PWRVEC= 24         ;;POWER FAIL
744      EMTVEC= 30         ;;EMULATOR TRAP (EMT) **ERROR**
745      TRAPVEC=34         ;;'TRAP' TRAP
746      TKVEC= 60          ;;TTY KEYBOARD VECTOR
747      TPVEC= 64          ;;TTY PRINTER VECTOR
748      PIRQVEC=240        ;;PROGRAM INTERRUPT REQUEST VECTOR
749
750
751
752      ADRS= R1
753      DLADDR= 176500
754
755      ; THE FOLLOWING DEFINITIONS APPLY TO THE GLOBAL SUBS
756      SET= -1
757      CLR= 0
758      TRUE= 1
  
```

```

759
760
761      ;:*****
762      ; RCSR REGISTER BIT NAMES
763      ;:*****
764      ; UNUSED      BIT15
765      ; UNUSED      BIT14
766      ; UNUSED      BIT13
767      ; UNUSED      BIT12
768      ; UNUSED      BIT11
769      ; UNUSED      BIT10
770      ; UNUSED      BIT09
771      ; UNUSED      BIT08
772      000200 RCVRDONE- BIT07      ; RECEIVER DONE
773      000100 RCVRIE-  BIT06      ; RECEIVER INTERRUPT ENABLE
774      ; UNUSED      BIT05
775      ; UNUSED      BIT04
776      ; UNUSED      BIT03
777      ; UNUSED      BIT02
778      ; UNUSED      BIT01
779      000001 RDRRUN=  BIT00      ; READER RUN
780
781      ;:*****
782      ; RBUF REGISTER BIT NAMES
783      ;:*****
784      100000 ERROR-      BIT15      ; ERROR INDICATOR
785      040000 ORERR=      BIT14      ; OVERRUN ERROR
786      C20000 FRERR=      BIT13      ; FRAMING ERROR
787      010000 PERR=      BIT12      ; PARITY ERROR
788      ; UNUSED      BIT11
789      ; UNUSED      BIT10
790      ; UNUSED      BIT09
791      ; UNUSED      BIT08
792      000200 RDATA7=      BIT07      ; \
793      000100 RDATA6=      BIT06      ; :
794      000040 RDATA5=      BIT05      ; /
795      000020 RDATA4=      BIT04      ; RECEIVED DATA BITS
796      000010 RDATA3=      BIT03      ;
797      000004 RDATA2=      BIT02      ;
798      000002 RDATA1=      BIT01      ;
799      000001 RDATA0=      BIT00      ; /
  
```

```

800
801
802
803
804
805
806
807
808
809
810
811
812      000200
813      000100
814
815
816
817
818
819      000001
820
821
822
823
824
825
826
827
828
829
830
831
832
833      000200
834      000100
835      000040
836      000020
837      000010
838      000004
839      000002
840      000001
841
842
843
844
845
846      000002
847      000004
848      000010
849      000020
  
```

```

;*****
; TCSR REGISTER BIT NAMES
;*****
; UNUSED      BIT15
; UNUSED      BIT14
; UNUSED      BIT13
; UNUSED      BIT12
; UNUSED      BIT11
; UNUSED      BIT10
; UNUSED      BIT09
; UNUSED      BIT08
XMITRDY=      BIT07      ; TRANSMITTER READY
XMITIE=       BIT06      ; TRANSMITTER INTERRUPT ENABLE
; UNUSED      BIT05
; UNUSED      BIT04
; UNUSED      BIT03
; UNUSED      BIT02
; UNUSED      BIT01
BREAK=        BIT00      ; SEND BREAK (CONTINUOUS SPACE)
  
```

```

;*****
; TBUF REGISTER BIT NAMES
;*****
; UNUSED      BIT15
; UNUSED      BIT14
; UNUSED      BIT13
; UNUSED      BIT12
; UNUSED      BIT11
; UNUSED      BIT10
; UNUSED      BIT09
; UNUSED      BIT08
TDATA7=       BIT07
TDATA6=       BIT06
TDATA5=       BIT05
TDATA4=       BIT04
TDATA3=       BIT03
TDATA2=       BIT02
TDATA1=       BIT01
TDATA0=       BIT00
; \
; \
; \ TRANSMITTER DATA BUFFER
; \
; \
; \
  
```

```

;*****
; FLAG BITS TO BE USE OR CLEARED IN SUSWR.
;*****
PARITY - 2
EVENODD - 4
BRK - 10
WRAP - 20
  
```

```

850
851
852
853
854          000000
855
856
857
858          000174
859 000174 000000
860 000176 000000
861
862 000200 000137 001334
863
864
865
866
867
868
869          000400
870
871 000400 005000
872 000402 012720 013610
873 000406 012720 000340
874 000412 020027 000400
875 000416 001371
876 000420 005037 000176
877 000424 000137 001334

;*****
;SMTL TRAP CATCHER
;
;      -0
; *ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
; *SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
; *LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
;
;      =174
DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
SWREG:   .WORD 0          ;;SOFTWARE SWITCH REGISTER
;SMTL STARTING ADDRESS(ES)
;      JMP @START ;;JUMP TO STARTING ADDRESS OF PROGRAM

; OFFLINE TESTING FOR INTERRUPT VECTOR PROBLEMS
;
;      =400
1$:      CLR      R0
;      MOV      #INTSRV,(R0)+ ;SET INTR HANDLER PTR
;      MOV      #PR7,(R0)+   ;SET PRIORITY
;      CMP      R0,#400      ;ALL DONE?
;      BNE      1$           ;BR IF NO
;      CLR      176          ;CLEAN UP SWREG
;      JMP      START        ;GO DO IT

```

```

878
879
880
881
882
883      000430
884      000046
885 000046 014450
886      000052
887 000052 000000
888      000430
889      001000
890
891
892
893
894
895      001000
896      000024
897 000024 000200
898      000044
899 000044 001000
900      001000
901
902
903
904
905 001000
906 001000 000000
907 001002 001174
908 001004 000031
909 001006 000144
910 001010 000144
911 001012 000030

      .SBTTL ACT11 HOOKS
      ;*****
      ;HOOKS REQUIRED BY ACT11
      $SVPC-      ;SAVE PC
      =46
      $ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
      =52
      .WORD 0      ;;2)SET LOC.52 TO ZERO
      -$SVPC      ;; RESTORE PC
      .1000
      .SBTTL APT PARAMETER BLOCK
      ;*****
      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
      ;*****
      .SX-      ;;SAVE CURRENT LOCATION
      .24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
      200      ;;FOR APT START UP
      .44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
      $APTHDR    ;;POINT TO APT HEADER BLOCK
      -.SX      ;;RESET LOCATION COUNTER
      ;*****
      ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
      ;INTERFACE SPEC.

$APTHD:
$HIBTS: .WORD 0      ;; TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL  ;; ADDRESS OF APT MAILBOX (BITS 0-15)
$TSTM: .WORD 25      ;; RUN TIM OF LONGEST TEST
$PASTM: .WORD 100.    ;; RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 100.    ;; ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
      .WORD $END-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
  
```

912  
 913  
 914  
 915  
 916  
 917  
 918 001100  
 919 001100  
 920 001100 000000  
 921 001102 000  
 922 001103 000  
 923 001104 000000  
 924 001106 000000  
 925 001110 000000  
 926 001112 000000  
 927 001114 000  
 928 001115 001  
 929 001116 000000  
 930 001120 000000  
 931 001122 000000  
 932 001124 000000  
 933 001126 000000  
 934 001130 000000  
 935 001132 000000  
 936 001134 000  
 937 001135 000  
 938 001136 000000  
 939 001140 177570  
 940 001142 177570  
 941 001144 177560  
 942 001146 177562  
 943 001150 177564  
 944 001152 177566  
 945 001154 000  
 946 001155 002  
 947 001156 012  
 948 001157 000  
 949 001160 000000  
 950 001162 000000  
 951 001164 177607 000377  
 952 001170 077  
 953 001171 015  
 954 001172 000012  
 955  
 956  
 957  
 958  
 959  
 960 001174  
 961 001174 000000  
 962 001176 000000  
 963 001200 000000  
 964 001202 000000  
 965 001204 000000  
 966 001206 000000  
 967 001210 000000

.SBTTL COMMON TAGS

\*\*\*\*\*  
 : THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS  
 : USED IN THE PROGRAM.

.=1100  
 \$CMTAG: :: START OF COMMON TAGS  
 .WORD 0  
 \$TSTNM: .BYTE 0 :: CONTAINS THE TEST NUMBER  
 \$ERFLG: .BYTE 0 :: CONTAINS ERROR FLAG  
 \$ICNT: .WORD 0 :: CONTAINS SUBTEST ITERATION COUNT  
 \$LPADR: .WORD 0 :: CONTAINS SCOPE LOOP ADDRESS  
 \$LPERR: .WORD 0 :: CONTAINS SCOPE RETURN FOR ERRORS  
 \$ERTIL: .WORD 0 :: CONTAINS TOTAL ERRORS DETECTED  
 \$ITEMB: .BYTE 0 :: CONTAINS ITEM CONTROL BYTE  
 \$ERMAX: .BYTE 1 :: CONTAINS MAX. ERRORS PER TEST  
 \$ERRPC: .WORD 0 :: CONTAINS PC OF LAST ERROR INSTRUCTION  
 \$GDADR: .WORD 0 :: CONTAINS ADDRESS OF 'GOOD' DATA  
 \$BDADR: .WORD 0 :: CONTAINS ADDRESS OF 'BAD' DATA  
 \$GDDAT: .WORD 0 :: CONTAINS 'GOOD' DATA  
 \$BDDAT: .WORD 0 :: CONTAINS 'BAD' DATA  
 .WORD 0 :: RESERVED--NOT TO BE USED  
 .WORD 0  
 \$AUTOB: .BYTE 0 :: AUTOMATIC MODE INDICATOR  
 \$INTAG: .BYTE 0 :: INTERRUPT MODE INDICATOR  
 .WORD 0  
 \$SWR: .WORD DSWR :: ADDRESS OF SWITCH REGISTER  
 \$DISPLAY: .WORD DDISP :: ADDRESS OF DISPLAY REGISTER  
 \$TKS: 177560 :: TTY KBD STATUS  
 \$TKB: 177562 :: TTY KBD BUFFER  
 \$TPS: 177564 :: TTY PRINTER STATUS REG. ADDRESS  
 \$TPB: 177566 :: TTY PRINTER BUFFER REG. ADDRESS  
 \$NULL: .BYTE 0 :: CONTAINS NULL CHARACTER FOR FILLS  
 \$FILLS: .BYTE 2 :: CONTAINS # OF FILLER CHARACTERS REQUIRED  
 \$FILLC: .BYTE 12 :: INSERT FILL CHARS. AFTER A 'LINE FEED'  
 \$TPFLG: .BYTE 0 :: 'TERMINAL AVAILABLE' FLAG (BIT<07> 0 YES)  
 \$TIMES: 0 :: MAX. NUMBER OF ITERATIONS  
 \$ESCAPE: 0 :: ESCAPE ON ERROR ADDRESS  
 \$BELL: .ASCIIZ <207><377><377> :: CODE FOR BELL  
 \$QUES: .ASCII /?/ :: QUESTION MARK  
 \$CRLF: .ASCII <15> :: CARRIAGE RETURN  
 \$LF: .ASCIIZ <12> :: LINE FEED

.SBTTL APT MAILBOX-ETABLE

\*\*\*\*\*  
 .EVEN  
 \$MAIL: :: APT MAILBOX  
 \$MSGTY: .WORD AMSGTY :: MESSAGE TYPE CODE  
 \$FATAL: .WORD AFATAL :: FATAL ERROR NUMBER  
 \$TESTN: .WORD ATESTN :: TEST NUMBER  
 \$PASS: .WORD APASS :: PASS COUNT  
 \$DEVCT: .WORD ADEVCT :: DEVICE COUNT  
 \$UNIT: .WORD AUNIT :: I/O UNIT NUMBER  
 \$MSGAD: .WORD AMSGAD :: MESSAGE ADDRESS

|      |        |        |                |        |                                                          |
|------|--------|--------|----------------|--------|----------------------------------------------------------|
| 968  | 001212 | 000000 | \$MSGLG: .WORD | AMSGLG | ::MESSAGE LENGTH                                         |
| 969  | 001214 |        | \$ETABLE:      |        | ::APT ENVIRONMENT TABLE                                  |
| 970  | 001214 | 000    | \$ENV: .BYTE   | AENV   | ::ENVIRONMENT BYTE                                       |
| 971  | 001215 | 000    | \$ENVM: .BYTE  | AENVM  | ::ENVIRONMENT MODE BITS                                  |
| 972  | 001216 | 000000 | \$SWREG: .WORD | ASWREG | ::APT SWITCH REGISTER                                    |
| 973  | 001220 | 001031 | \$USWR: .WORD  | AUSWR  | ::USER SWITCHES                                          |
| 974  | 001222 | 000000 | \$CPUOP: .WORD | ACPUOP | ::CPU TYPE,OPTIONS                                       |
| 975  |        |        | ::*            |        | BITS 15-11=CPU TYPE                                      |
| 976  |        |        | ::*            |        | 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05             |
| 977  |        |        | ::*            |        | 11/70=06,PDQ=07,Q=10                                     |
| 978  |        |        | ::*            |        | BIT 10=REAL TIME CLOCK                                   |
| 979  |        |        | ::*            |        | BIT 9=FLOATING POINT PROCESSOR                           |
| 980  |        |        | ::*            |        | BIT 8=MEMORY MANAGEMENT                                  |
| 981  | 001224 | 000    | \$MAMS1: .BYTE | AMAMS1 | ::HIGH ADDRESS,M.S. BYTE                                 |
| 982  | 001225 | 000    | \$MTYP1: .BYTE | AMTYP1 | ::MEM. TYPE,BLK#1                                        |
| 983  |        |        | ::*            |        | MEM.TYPE BYTE -- (HIGH BYTE)                             |
| 984  |        |        | ::*            |        | 900 NSEC CORE=001                                        |
| 985  |        |        | ::*            |        | 300 NSEC BIPOLAR=002                                     |
| 986  |        |        | ::*            |        | 500 NSEC MOS=003                                         |
| 987  | 001226 | 000000 | \$MADR1: .WORD | AMADR1 | ::HIGH ADDRESS,BLK#1                                     |
| 988  |        |        | ::*            |        | MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE |
| 989  | 001230 | 000    | \$MAMS2: .BYTE | AMAMS2 | ::HIGH ADDRESS,M.S. BYTE                                 |
| 990  | 001231 | 000    | \$MTYP2: .BYTE | AMTYP2 | ::MEM. TYPE,BLK#2                                        |
| 991  | 001232 | 000000 | \$MADR2: .WORD | AMADR2 | ::MEM.LAST ADDRESS,BLK#2                                 |
| 992  | 001234 | 000    | \$MAMS3: .BYTE | AMAMS3 | ::HIGH ADDRESS,M.S.BYTE                                  |
| 993  | 001235 | 000    | \$MTYP3: .BYTE | AMTYP3 | ::MEM. TYPE,BLK#3                                        |
| 994  | 001236 | 000000 | \$MADR3: .WORD | AMADR3 | ::MEM.LAST ADDRESS,BLK#3                                 |
| 995  | 001240 | 000    | \$MAMS4: .BYTE | AMAMS4 | ::HIGH ADDRESS,M.S.BYTE                                  |
| 996  | 001241 | 000    | \$MTYP4: .BYTE | AMTYP4 | ::MEM. TYPE,BLK#4                                        |
| 997  | 001242 | 000000 | \$MADR4: .WORD | AMADR4 | ::MEM.LAST ADDRESS,BLK#4                                 |
| 998  | 001244 | 000300 | \$VECT1: .WORD | AVECT1 | ::INTERRUPT VECTOR#1,BUS PRIORITY#1                      |
| 999  | 001246 | 000000 | \$VECT2: .WORD | AVECT2 | ::INTERRUPT VECTOR#2BUS PRIORITY#2                       |
| 1000 | 001250 | 176500 | \$BASE: .WORD  | ABASE  | ::BASE ADDRESS OF EQUIPMENT UNDER TEST                   |
| 1001 | 001252 | 000000 | \$DEVN: .WORD  | ADEVN  | ::DEVICE MAP                                             |
| 1002 | 001254 |        | \$ETEND:       |        |                                                          |
| 1003 |        |        | .MEXIT         |        |                                                          |

1004  
 1005  
 1006  
 1007  
 1008  
 1009  
 1010  
 1011  
 1012  
 1013  
 1014  
 1015  
 1016  
 1017  
 1018 001254  
 1019  
 1020 001254 176500  
 1021 001256 000300  
 1022 001260 176500  
 1023 001262 176502  
 1024 001264 176504  
 1025 001266 176506  
 1026 001270 000000  
 1027 001272 000020  
 1028 001332 000000

.SBTTL ERROR POINTER TABLE

;\*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.  
 ;\*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN  
 ;\*LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.  
 ;\*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).  
 ;\*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

;\* EM ;;POINTS TO THE ERROR MESSAGE  
 ;\* DH ;;POINTS TO THE DATA HEADER  
 ;\* DT ;;POINTS TO THE DATA  
 ;\* DF ;;POINTS TO THE DATA FORMAT

\$ERRTB:

;; GLOBAL DATA  
 DLADD: DLADDR  
 DLVEC: 300  
 RCSR: DLADDR + 0  
 RBUF: DLADDR + 2  
 TCSR: DLADDR + 4  
 TBUF: DLADDR + 6  
 I: 0  
 .BLKW 20 :FOR R5 STACK  
 RSSTACK: .WORD 0

```

1029 001334
1030
1031
1032 001334 012706 001100
1033 001340 005026
1034 001342 022706 001140
1035 001346 001374
1036 001350 012706 001100
1037
1038 001354 012737 016210 000020
1039 001362 012737 000340 000022
1040 001370 012737 016010 000030
1041 001376 012737 000340 000032
1042 001404 012737 017142 000034
1043 001412 012737 000340 000036
1044 001420 013737 014416 014410
1045 001426 005037 001160
1046 001432 005037 001162
1047 001436 112737 000001 001115
1048 001444 012737 001444 001106
1049 001452 012737 001452 001110
1050
1051
1052 001460 013746 000004
1053 001464 012737 001520 000004
1054 001472 012737 177570 001140
1055 001500 012737 177570 001142
1056 001506 022777 177777 177424
1057 001514 001012
1058
1059 001516 000403
1060 001520 012716 001526
1061 001524 000002
1062 001526 012737 000176 001140
1063 001534 012737 000174 001142
1064 001542 012637 000004
1065
1066 001546 005037 001202
1067 001552 132737 000200 001215
1068 001560 001403
1069 001562 012737 001216 001140
1070 001570
1071
1072
1073 001570 005227 177777
1074 001574 001043
1075 001576 022737 014450 000042
1076 001604 001437
1077 001606 104401 001654
1078
1079 001612 005737 000042
1080 001616 001012
1081 001620 123727 001214 000001
1082 001626 001406
1083 001630 023727 001140 000176
1084 001636 001005

START:
.SBTTL INITIALIZE THE COMMON TAGS
;;CLEAR THE COMMON TAGS ($CMTAG) AREA
MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
CLR (R6)+ ;;CLEAR MEMORY LOCATION
CMP #SWR,R6 ;;DONE?
BNE -6 ;;LOOP BACK IF NO
MOV #STACK,SP ;;SETUP THE STACK POINTER
;;INITIALIZE A FEW VECTORS
MOV #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
MOV #340,@IOTVEC+2 ;;LEVEL 7
MOV #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
MOV #340,@EMTVEC+2 ;;LEVEL 7
MOV #TRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
MOV #340,@TRAPVEC+2 ;;LEVEL 7
MOV $ENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
MOV #1,$ERMAX ;;ALLOW ONE ERROR PER TEST
MOV #,$SLPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
MOV #,$SLPERR ;;SETUP THE ERROR LOOP ADDRESS
;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
MOV @ERRVEC,-(SP) ;;SAVE ERROR VECTOR
MOV #64,$@ERRVEC ;;SET UP ERROR VECTOR
MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
;;AND THE HARDWARE SWR IS NOT -1
BR 65$ ;;BRANCH IF NO TIMEOUT
64$: MOV #65$,(SP) ;;SET UP FOR TRAP RETURN
65$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
MOV #DISPREG,DISPLAY
66$: MOV (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR
CLR $PASS ;;CLEAR PASS COUNT
BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
BEQ 67$ ;;YES,USE NON-APT SWITCH
MOV #SWREG,SWR ;;NO,USE APT SWITCH REGISTER
67$:
.SBTTL TYPE PROGRAM NAME
;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
INC #-1 ;;FIRST TIME?
BNE 68$ ;;BRANCH IF NO
CMP #SENDAD,@#42 ;;ACT-11?
BEQ 68$ ;;BRANCH IF YES
TYPE ,69$ ;;TYPE ASCII STRING
.SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
BNE 70$ ;;BRANCH IF YES
CMPB $ENV,#1 ;;ARE WE RUNNING UNDER APT?
BEQ 70$ ;;BRANCH IF YES
CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
BNF 71$ ;;BRANCH IF NO

```

N 2  
CVDLAA0 DLV11-J TEST MACY11 30G(1063) 17-NOV-78 09:39 PAGE 27  
CVDLAB.P11 17-NOV-78 09:38 GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0026

|      |        |        |        |        |         |        |                                    |                           |
|------|--------|--------|--------|--------|---------|--------|------------------------------------|---------------------------|
| 1085 | 001640 | 104406 |        |        |         | GTSWR  |                                    | ::GET SOFT-SWR SETTINGS   |
| 1086 | 001642 | 000403 |        |        |         | BR     | 71\$                               |                           |
| 1087 | 001644 | 112737 | 000001 | 001134 | 70\$:   | MOVB   | #1,\$AUTOB                         | ::SET AUTO-MODE INDICATOR |
| 1088 | 001652 |        |        |        | 7\$:    |        |                                    |                           |
| 1089 | 001652 | 000414 |        |        |         | BR     | 68\$                               | ::GET OVER THE ASCIZ      |
| 1090 |        |        |        |        | ::69\$: | .ASCIZ | <CRLF>*CVDLAB0 DLV11-J TEST*<CRLF> |                           |
| 1091 | 001704 |        |        |        | 68\$:   |        |                                    |                           |

```
1092
1093
1094 001704 004737 012700 JSR PC,SIZE ;SIZE THE BUS IF $DEVM = 0
1095
1096 001710 005037 014166 CLR PHASE2 ;INIT FOR TESTS
1097 001714 012737 000001 001206 MOV #1,$UNIT
1098 001722 005037 014164 CLR CHAN
1099 001726 005037 014170 CLR P1CNT
1100
1101 001732 012706 001100 LOOP: MOV #STACK,SP ;RESET STACK POINTER
1102 001736 004737 013620 JSR PC,CYCLE ;SETUP MODULE AND CHANNEL
1103 ;
1104 ;ADRS+2 -> NEXT VECTOR
1105 001742 LET ;GET UNIT ADDRESS
1106 001742 012137 001254 MOV (ADRS)+,DLADD DLADD : (ADRS)+
1107 ;GET UNIT VECTOR
1108 001746 LET DLVEC := (ADRS)
1109 001746 011137 001256 MOV (ADRS),DLVEC
1110 ;RCSR = DLADD + 0
1111 001752 LET RCSR := DLADD
1112 001752 013737 001254 001260 MOV DLADD,RCSR
1113 001760 LET RBUF := DLADD + #2
1114 001760 013737 001254 001262 MOV DLADD,RBUF
1115 001766 062737 000002 001262 ADD #2,RBUF
1116 001774 LET TCSR : DLADD + #4
1117 001774 013737 001254 001264 MOV DLADD,TCSR
1118 002002 062737 000004 001264 ADD #4,TCSR
1119 002010 LET TBUF : DLADD + #6
1120 002010 013737 001254 001266 MOV DLADD,TBUF
1121 002016 062737 000006 001266 ADD #6,TBUF
1122 002024 LET R5 : #R5STACK
1123 002024 012705 001332 MOV #R5STACK,R5
1124
1125 002030 012700 001000 MOV #1000,R0 ;DELAY TO ALLOW PRINT TO FINISH
1126 002034 077001 SOB R0,, ;BEFORE RESET
1127 002036 BRESET
1128 002036 000005 RESET
1129 002040 LET $DEVCT : $DEVCT + #1
1130 002040 005237 001204 INC $DEVCT
1131 002044 IFB PHASE2 EQ #TRUE THEN
1132 002044 123727 014166 000001 CMPB PHASE2,#TRUE
1133 002052 001003 BNE $1
1134 002054 000137 007000 JMP MODTST ;GO TO PHASE 2 TESTING
1135 002060 ELSE
1136 002060 000402 BR $2
1137 002062 $1:
1138 002062 LET $TSTNM :- #0
1139 002062 005037 001102 CLR $TSTNM
1140 002066 ENDIF
1141 002066 $2:
```

```

1142 *****
1143 :*TEST 1 ADDRESSABILITY
1144 :*
1145 :* THIS TEST VERIFIES THAT ALL 4 REGISTERS OF THE CHANNEL UNDER TEST
1146 :* RESPOND TO THEIR ADDRESSES.
1147 :*
1148 :*****
1149 TST1: SCOPE
1150 002066 000004 MOV #2,$TIMES ;;DO 2 ITERATIONS
1151 002070 012737 000002 001160 MOV #1,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1152 002076 012737 000001 001200 LET ADRS := DLADD
1153 002104 013701 001254 MOV DLADD,ADRS
1154 ; SET UP INTERRUPT
1155 002110 012737 013610 000004 MOV #INTSRV,ERRVEC
1156 002116 012737 000340 000006 MOV #PR7,ERRVEC+2
1157 002124 CLR I LET I := #0
1158 002124 005037 001270 REPEAT
1159 002130 $3: BGNSUB
1160 002130 MOV #64,$,SLPERR
1161 002130 012737 002136 001110 ;CLEAR FLAG
1162 ; LET INTFLAG : #0
1163 002136 005037 013616 CLR INTFLAG
1164 ;READ FLAG
1165 002142 005711 TST @ADRS IF INTFLAG NE #0 THEN
1166 002144 005737 013616 TST INTFLAG
1167 002150 001401 BEQ $4 ; FATAL ERROR
1168 ; ERRDF 1
1169 002152 104001 ERROR 1
1170 002154 $4: ENDF
1171 002154 ENDSUB
1172 002154 LET I := I + #2
1173 002154 062737 000002 001270 LET ADRS := DLADD + I
1174 002162 013701 001254 MOV DLADD,ADRS
1175 002166 063701 001270 ADD I,ADRS
1176 002172 023727 001270 000010 CMP I,#8.
1177 002200 001353 BNE $3 UNTIL I EQ #8.

```



```

1206 .....
1207 * THE FOLLOWING 3 TESTS TEST ALL 'READ WRITE' BITS
1208 .....
1209 .....
1210 .....
1211 .....
1212 *TEST 2 BREAK - TCSR 0 SET, CLEAR, RESET
1213 * THIS BIT IS THE ONLY ONE IN THIS POSITION
1214 * THAT IS READ AND WRITE.
1215 .....
1216 TST2: SCOPE
1217 002264 000004 MOV #10,$TIMES ;;DO 10 ITERATIONS
1218 002266 012737 0000*0 001160 MOV #2,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1219 002274 012737 000002 001200
1220 IF #BRK NOTSETIN $USWR THEN
1221 002302 032737 000010 001220 BIT #BRK,$USWR
1222 002310 001001 BNE $5
1223 002312 EXIT ; BREAK NOT INSTALLED
1224 002312 000452 BR TST3 ;;EXIT THIS TEST
1225 002314 ENDIF
1226 002314 $5:
1227
1228 ; SEE IF IT IS CLEAR
1229 002314 BGNSUB
1230 002314 012737 002322 001110 MOV #64,$LPERR
1231
1232 IF #BREAK SETIN @TCSR THEN
1233 002322 032777 000001 176734 BIT #BREAK,@TCSR
1234 002330 001401 BEQ $6
1235 ; BREAK DID NOT RESET IN TCSR
1236 002332 ERRHRD 2
1237 002332 104002 ERROR 2
1238 002334 ENDIF
1239 002334 $6:
1240 002334 ENDSUB
1241
1242 ; TRY TO SET BREAK BIT
1243 002334 BGNSUB
1244 002334 012737 002342 001110 MOV #64,$LPERR
1245 002342 LET @TCSR := @TCSR SET.BY #BREAK
1246 002347 052777 000001 176714 BIS #BREAK,@TCSR
1247 ; STUCK TO 0
1248 002350 IF #BREAK NOTSETIN @TCSR THEN
1249 002350 032777 000001 176706 BIT #BREAK,@TCSR
1250 002356 001001 BNE $7
1251 ; BREAK DID NOT SET IN TCSR
1252 002360 ERRHRD 3
1253 002360 104003 ERROR 3
1254 002362 ENDIF
1255 002362 $7:
1256 002362 ENDSUB
1257
1258 ; TRY TO CLEAR A SET BIT
1259 002362 BGNSUB
1260 002362 012737 002370 001110 MOV #0,$LPERR
1261

```

```

262 002370          LET @TCSR := @TCSR CLR.BY #BREAK
1263 002370 042777 000001 176666 BIC #BREAK,@TCSR
1264          ; SHOULD HAVE CLEARED
1265 002376          IF #BREAK SETIN @TCSR THEN
1266 002376 032777 000001 176660 BIT #BREAK,@TCSR
1267 002404 001401 BEQ $10
1268          ; BREAK DID NOT CLEAR IN TCSR
1269 002406          ERRHRD 4
1270 002406 104004 ERROR 4
1271 002410          ENDIF
1272 002410          $10:
1273 002410          ENDSUB
1274          ; NOW SEE IF RESET CLEARS IT
1275          BGNSUB
1276 002410          MOV #64$,$LPERR
1277 002410 012737 002416 0011'0 MOV #64$,$LPERR
1278          LET @TCSR := @TCSR SET.BY #BREAK
1279 002416          ; ISSUE BUS RESET
1280 002416 052777 000001 176640 BIS #BREAK,@TCSR
1281          BRESÉT
1282 002424          IF #BREAK SETIN @TCSR THEN
1283 002424 000005 RESET
1284 002426          ; BREAK DID NOT RESET IN TCSR
1285 002426 032777 000001 176630 BIT #BREAK,@TCSR
1286 002434 001401 BEQ $11
1287          ERRHRD 5
1288 002436          ERROR 5
1289 002436 104005 ERROR 5
1290 002440          ENDIF
1291 002440          $11:
1292 002440          ENDSUB
  
```

```

293
1294
1295
1296
1297 002440 000004
1298 002442 012737 000010 001160
1299 002450 012737 000003 001200
1300
1301 002456
1302
1303
1304 002470
1305 002470 012737 002476 001110
1306
1307 002476
1308 002476 032777 000100 176560
1309 002504 001401
1310
1311 002506
1312 002506 104012
1313 002510
1314 002510
1315 002510
1316
1317
1318 002510
1319 002510 012737 002516 001110
1320 002516
1321 002516 052777 000100 176540
1322
1323 002524
1324 002524 032777 000100 176532
1325 002532 001001
1326
1327 002534
1328 002534 104013
1329 002536
1330 002536
1331 002536
1332
1333
1334 002536
1335 002536 012737 002544 001110
1336
1337 002544
1338 002544 042777 000100 176512
1339
1340 002552
1341 002552 032777 000100 176504
1342 002560 001401
1343
1344 002562
1345 002562 104014
1346 002564
1347 002564
1348 002564

*****
*TEST 3          XMITIE - TCSR 6          SET, CLEAR, RESET
*****
TST3:  SCOPE
      MOV      #10,$TIMES          ;;DO 10 ITERATIONS
      MOV      #3,$TESTN          ;;SET TEST NUMBER IN APT MAIL BOX
                                   ; USE PRIORITY OF 7
      SETPRI  #PR7
                                   ; SEE IF IT IS CLEAR
                                   BGNSUB
      MOV      #65,$LPERR
      BIT      #XMITIE,@TCSR
      BEQ      $12
                                   ; XMITIE DID NOT RESET IN TCSR
                                   ERRHRD 12
      ERROP    12
                                   ENDIF
$12:
                                   ENDSUB
                                   ; TRY TO SET XMITIE BIT
                                   BGNSUB
      MOV      #64,$LPERR
      BIS      #XMITIE,@TCSR
      LET      @TCSR : @TCSR SET BY #XMITIE
                                   ; STUCK TO 0
      IF      #XMITIE NOTSETIN @TCSR THEN
                                   ; XMIT DID NOT SET IN TCSR
                                   ERRHRD 13
      ERROR    13
                                   ENDIF
$13:
                                   ENDSUB
                                   ; TRY TO CLEAR A SET BIT
                                   BGNSUB
      MOV      #64,$LPERR
      BIC      #XMITIE,@TCSR
      LET      @TCSR : @TCSR CLR BY #XMITIE
                                   ; SHOULD HAVE CLEARED
      IF      #XMITIE SETIN @TCSR THEN
                                   ; XMIT DID NOT CLEAR IN TCSR
                                   ERRHRD 14
      ERROR    14
                                   ENDIF
$14:
                                   ENDSUB

```

```

1349
1350 ; NOW SEE IF RESET CLEARS IT
1351 002564 012737 002572 001110 MOV #64$,SLPERR BGNSUB
1352 002564
1353
1354 002572 052777 000100 176464 BIS #XMITIE,@TCSR LET @TCSR := @TCSR SET.BY #XMITIE
1355 002572
1356 ; ISSUE BUS RESET
1357 002600 000005 RESET BRESET
1358 002600
1359 002602 032777 000100 176454 BIT #XMITIE,@TCSR IF #XMITIE SET IN @TCSR THEN
1360 002602
1361 002610 001401 BEQ $15 ; XMIT DID NOT RESET IN TCSR
1362 ERRHRD 15
1363 002612 ERROR 15
1364 002612 104015
1365 002614
1366 002614 $15:
1367 002614
ENDSUB

```

```

1368
1369
1370
1371
1372
1373
1374 002614 000004
1375 002616 012737 000010 001160
1376 002624 012737 000004 001200
1377
1378 002632
1379 002632 012737 002640 001110
1380
1381 002640
1382 002640 032777 000100 176412
1383 002646 001401
1384
1385 002650
1386 002650 104035
1387 002652
1388 002652
1389 002652
1390
1391
1392 002652
1393 002652 012737 002660 001110
1394 002660
1395 002660 052777 000100 176372
1396
1397 002666
1398 002666 032777 000100 176364
1399 002674 001001
1400
1401 002676
1402 002676 104036
1403 002700
1404 002700
1405 002700
1406
1407
1408 002700
1409 002700 012737 002706 001110
1410
1411 002706
1412 002706 042777 000100 176344
1413
1414 002714
1415 002714 032777 000100 176336
1416 002722 001401
1417
1418 002724
1419 002724 104037
1420 002726
1421 002726
1422 002726
1423

*****
TEST 4          RCVRIE - RCSR 6          SET, CLEAR, RESET
                THIS BIT IS THE ONLY ONE IN THIS POSITION
                THAT IS READ AND WRITE.
*****
TST4:  SCOPE
        MOV      #10,$TIMES          ;;DO 10 ITERATIONS
        MOV      #4,$TESTN          ;;SET TEST NUMBER IN APT MAIL BOX
                ; SEE IF IT IS CLEAR
                BGNSUB
        MOV      #64$,$LPERR
        IF      #RCVRIE SETIN @RCSR THEN
        BIT     #RCVRIE,@RCSR
        BEQ     $16
                ; RCVRIE DID NOT RESET IN RCSR
                ERRHRD 35
        ENDIF
        $ 5:
        ENDSUB
                ; TRY TO SET RCVRIE BIT
        BGNSUB
        MOV      #64$,$LPERR
        LET      @RCSR : @RCSR SET.BY #RCVRIE
        BIS      #RCVRIE,@RCSR
                ; STUCK TO 0
        IF      #RCVRIE NOTSETIN @RCSR THEN
                ; RCVRIE DID NOT SET IN RCSR
                ERRHRD 36
        ENDIF
        $17:
        ENDSUB
                ; TRY TO CLEAR A SET BIT
        BGNSUB
        MOV      #64$,$LPERR
        LET      @RCSR := @RCSR CLR.BY #RCVRIE
        BIC      #RCVRIE,@RCSR
                ; SHOULD HAVE CLEARED
        IF      #RCVRIE SETIN @RCSR THEN
                ; RCVRIE DID NOT CLEAR IN RCSR
                ERRHRD 37
        ENDIF
        $20:
        ENDSUB
  
```

```

(VDLAAO DLV11-J TEST MACY11 30G(1063) 17-NOV-78 09:39 PAGE 36 3
(VDLAB.P11 17-NOV-78 09:38 T4 RCVRIE - RCSR 6 SET, CLEAR, RESET SEL 0035

; NOW SEE IF RESET CLEARS IT
BGNSUB
1424
1425 002726
1426 002726 012737 002734 001110 MOV #64$, $LPERR
1427
1428 002734 LET @RCSR := @RCSR SET BY #RCVRIE
1429 002734 052777 000100 176316 BIS #RCVRIE, @RCSR
1430 ; ISSUE BUS RESET
1431 002742 BRES1T
1432 002742 000005 RESET
1433 002744 IF #RCVRIE SET IN @RCSR THEN
1434 002744 032777 000100 176306 BIT #RCVRIE, @RCSR
1435 002752 001401 BEQ $21
1436 ; RCVRIE DID NOT RESET IN RCSR
1437 002754 ERRHRD 40
1438 002754 104040 ERROR 40
1439 002756 ENDF
1440 002756
1441 002756 S21: ENDSUB

```

```

1442
1443
1444
1445
1446 002756 000004
1447 002760 012737 000010 001160
1448 002766 012737 000005 001200
1449
1450 002774
1451 002774 012737 003002 001110
1452
1453 003002
1454 003002 032777 000200 176254
1455 003010 001002
1456
1457
1458
1459 003012
1460 003012 104042
1461
1462 003014
1463 003014 000005
1464 003016
1465 003016
1466
1467 003016

*****
:TEST 5 TEST THAT XMITRDY - TCSR 7 - IS SET BY INIT
*****
TST5: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #5,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
BGNSUB
MOV #64$,$LPERR
IF #XMITRDY NOTSETIN @TCSR THEN
BIT #XMITRDY,@TCSR
BNE $22
;RESET SHOULD HAVE SET BIT.
;XMITRDY DID NOT SET IN TCSR (AFTER RESE
ERRHRD 42
;ISSUE ANOTHER RESET
BRESET
ENDIF
;ALLOW LOOPING ON ERROR
ENDSUB
$22:
  
```

```

1468
1469
1470
1471
1472
1473
1474 003016 000004
1475 003020 012737 000010 001160
1476 003026 012737 000006 001200
1477
1478 003034
1479 003034 023727 013604 000001
1480 003042 001001
1481 003044
1482 003044 000521
1483 003046
1484 003046
1485
1486 003046
1487 003046 005037 003302
1488 003052
1489 003052
1490
1491 003052
1492 003052 005037 003304
1493 003056
1494 003056 005037 003306
1495
1496
1497
1498
1499
1500
1501 003062
1502 003062 105077 176200
1503
1504
1505
1506 003066
1507 003066 010546
1508 003070 012745 177777
1509 003074 013745 001264
1510 003100 012745 000200
1511 003104 012745 000500
1512 003110 004737 012446
1513 003114 012605
1514
1515
1516 003116
1517 003116 103003
1518
1519 003120
1520 003120 104066
1521 003122 000137 003310
1522 003126
1523 003126

*****
*TEST 6          TEST THAT XMIT RDY - TCSR 7 - CLEARS
*                WHEN TBUF IS LOADED WITH A CHARACTER
*                AND THAT IT SETS WITHIN A REASONABLE AMOUNT OF TIME.
*****
TST6:  SCOPE
      MOV  #10,$TIMES      ;;DO 10 ITERATIONS
      MOV  #6,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX

                               IF CONSOL EQ #TRUE THEN
      CMP  CONSOL,#TRUE
      BNE  $23
                               EXIT
      BR   TST7              ;;EXIT THIS TEST
                               ENDIF

$23:
                               LET PASS := #0 ;INIT COUNT OF TIMES THRU
                               LOOP          ; START OF LOOP
                               ; MAX OF 2 TIMES THRU
                               LET ERRORFLAG : #0
                               LET EXITFLAG : #0
      CLR  ERRORFLAG
      CLR  EXITFLAG
      ; LOAD TBUF WITH ONE CHARACTER
      ; WAIT FOR READY TO SET
      ; (SHOULD BE VERY SHORT WAIT
      ; SINCE UART DOUBLE BUFFERS ITS INPUT)
      ; SEND A CHARACTER
      LET @TBUF :B- #0
      CLRB @TBUF
      ;WAIT A MAXIMUM
      ;OF 50 MSEC FOR
      ;XMIT RDY TO SET IN TCSR
      CALL TIMER IN <#500,#XMITRDY,TCSR,#SET>

      MOV  R5,-(SP)
      MOV  #SET,-(R5)
      MOV  TCSR,-(R5)
      MOV  #XMITRDY,-(R5)
      MOV  #500,-(R5)
      JSR  PC,TIMER
      MOV  (SP)+,R5

      ;TIMER RETURNS AN ERROR IF BIT DID
      ;NOT MEET CONDITION WITHIN TIME LIMIT
      IF .ERROR THEN
      ;XMIT RDY DID NOT SET IN TCSR
      ERRHRD 66

      BCC  $26
      ERROR 66
      JMP  TST7
      ENDIF

$26:

```

```

1524 003126 105077 176134      1$: CLRB @TBUF ;SHIP 1'ST CHAR
1525 003132 105777 176126      TSTB @TCSR ;WAIT FOR RDY
1526 003136 100375      BPL 1$
1527      ; LOAD TBUF WITH A SECOND CHARACTER (TO DOUBLE BUFFER)
1528      ; CHECK IMMEDIATELY THAT XMITRDY IS CLEAR
1529      ; AND THEN WAIT FOR IT TO SET
1530
1531      ;SEND SECOND CHARACTER
1532      LET @TBUF :B= #0
1533 003140
1534 003140 105077 176122      CLRB @TBUF
1535 003144 000240      NOP
1536      ; GIVE IT TIME TO CLEAR
1537      ; XMITRDY SHOULD HAVE CLEARED UPON
1538      ; RECEIPT OF A CHARACTER
1539 003146 032777 000200 176110      BIT #XMITRDY,@TCSR
1540 003154 001404      BEQ $27
1541      ; XMITRDY DID NOT CLEAR IN TCSR
1542      ; WILL RESULT IN ERR 67 IF FAILS 2X
1543 003156 012737 177777 003304      MOV #SET,ERRORFLAG
1544      LET ERRORFLAG := #SET
1545      ; DEFER ERROR TYPEOUT
1546 003164
1547 003164 000416      BR $30
1548 003166      ELSE
1549      $27:
1550      ;WAIT A MAXIMUM
1551      ;OF 100 MSEC FOR
1552      ;XMIT RDY TO SET IN TCSR
1553 003166 010546      MOV R5,-(SP)
1554 003170 012745 177777      MOV #SET,-(R5)
1555 003174 013745 001264      MOV TCSR,-(R5)
1556 003200 012745 000200      MOV #XMITRDY,-(R5)
1557 003204 012745 001000      MOV #1000,-(R5)
1558 003210 004737 012446      JSR PC,TIMER
1559 003214 012605      MOV (SP)+,R5
1560 003216
1561 003216 103001      BCC $31
1562      IF.ERROR THEN
1563 003220      ;XMIT RDY DID NOT SET IN TCSR
1564 003220 104070      ERRHRD 70
1565 003222      ENDIF
1566 003222      $31:
1567 003222      $30:
1568 003222
1569 003222
1570 003222 023727 003304 177777      CMP ERRORFLAG,#SET
1571 003230 001013      BNE $32
1572 003232
1573 003232 005237 003302      INC PASS
1574 003236
1575 003236 023727 003302 000001      CMP PASS,#1
1576 003244 003404      BLE $33
1577      LET PASS := PASS + #1
1578      IF PASS GT #1 THEN
1579 003246      ; CALL ERROR IF 2ND TRY
1580      ; ON XMIT RDY NOT CLEARING
1581      ERRHRD 67

```

```

1580 003246 104067          ERROR 67
1581 003250                      LET EXITFLAG := #SET
1582 003250 012737 177777 003306  MOV  #SET,EXITFLAG
1583 003256                      ENDIF
1584 003256          $33:                      ELSE          ; NO ERROR
1585 003256                      BR          $34
1586 003256 000403          $32:
1587 003260                      LET EXITFLAG := #SET
1588 003260                      MOV  #SET,EXITFLAG
1589 003260 012737 177777 003306  ENDIF
1590 003266          $34:
1591 003266                      EXIF          EXITFLAG EQ #SET
1592 003266                      CMP  EXITFLAG,#SET
1593 003266 023727 003306 177777  BEQ  $25
1594 003274 001401                      ENDLOOP
1595 003276          BR          $24
1596 003276 000665          $25:
1597 003300                      EXIT ; SKIP AROUND FLAG WORDS
1598 003300                      ;;EXIT THIS TEST
1599 003300 000403          BR          TST7
1600 003302 000000          PASS: 0
1601 003304 000000          ERRORFLAG: 0
1602 003306 000000          EXITFLAG: 0

```

```
1603
1604
1605 *****
1606 *TEST 7      TEST THAT OUTPUTTING A CHAR FROM TBUF (WITH WRAP CONNECTED)
1607 *            RESULTS IN RCVRDONE SETTING WITHIN A REASONABLE AMOUNT OF TIME
1608 *            AND THAT RESET CLEARS THE BIT.
1609 *****
1610 003310 000004 TST7: SCOPE
1611 003312 012737 000010 001160 MOV #10,$TIMES ;;DO 10 ITERATIONS
1612 003320 012737 000007 001200 MOV #7,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1613
1614 003326 032737 000020 001220 BIT #WRAP,$USWR IF #WRAP NOTSET IN $USWR OR CONSOL EQ #TRUE THEN
1615 003334 001404 BEQ $35
1616 003336 023727 013604 000001 CMP CONSOL,#TRUE
1617 003344 001001 BNE $36
1618 003346 $35:
1619 003346 000434 BR TST10 EXIT
1620 003346 000434 ;;EXIT THIS TEST
1621 003350 ENDIF
1622 003350 $36:
1623
1624 003350 BGNSUB
1625 003350 012737 003356 001110 MOV #64,$LPERR
1626 ; SEND A CHARACTER AND LET IT WRAP AROUND
1627
1628 003356 LET @TBUF :B= #0
1629 003356 105077 175704 CLRB @TBUF
1630
1631 ; WAIT A MAXIMUM OF 50 MSEC
1632 ; FOR RCVR DONE TO SET IN
1633 ; RCSR
1634 003362 CALL TIMER IN <#500,#RCVRDONE,RCSR,#SET>
1635 003362 010546 MOV R5,-(SP)
1636 003364 012745 177777 MOV #SET,-(R5)
1637 003370 013745 001260 MOV RCSR,-(R5)
1638 003374 012745 000200 MOV #RCVRDONE,-(R5)
1639 003400 012745 000500 MOV #500,-(R5)
1640 003404 004737 012446 JSR PC,TIMER
1641 003410 012605 MOV (SP)+,R5
1642
1643 ;DIDN'T SET IN TIME
1644 003412 IF .ERROR THEN
1645 003412 103001 BCC $37
1646
1647 003414 ; RCVRDONE DID NOT SET IN RCSR
1648 003414 104071 ERROR 71
1649 003416 ERRHRD 71
1650 003416 $37: ENDF
1651
1652 003416 ENDSUB
1653
1654 003416 BGNSUB
1655 003416 012737 003424 001110 MOV #64,$LPERR
1656 ; NOW THAT IT IS SET SEE IF IT CAN BE RESET
1657 003424 BRESET
1658 003424 000005 RESET
```

```
1659
1660 003426
1661 003426 032777 000200 175624 BIT #RCVRDONE,@RCSR
1662 003434 001401 BEQ $40
1663 ; RCVRDONE DID NOT RESET IN RCSR.
1664 003436 ERRHRD 72
1665 003436 104072 ERROR 72
1666 003440
1667 003440 $40:
1668 003440 ENDSUB
```

```

1669
1670
1671
1672
1673 003440 000004
1674 003442 012737 000010 001160
1675 003450 012737 000010 001200
1676
1677 003456
1678 003456 032737 000020 001220
1679 003464 001404
1680 003466 023727 013604 000001
1681 003474 001001
1682 003476
1683 003476
1684 003476 000433
1685 003500
1686 003500
1687
1688 003500
1689 003500 012737 003506 001110
1690
1691
1692
1693 003506
1694 003506 105077 175554
1695
1696
1697
1698 003512
1699 003512 010546
1700 003514 012745 177777
1701 003520 013745 001260
1702 003524 012745 000200
1703 003530 012745 000500
1704 003534 004737 012446
1705 003540 012605
1706
1707 003542
1708 003542 103001
1709
1710 003544
1711 003544 104025
1712
1713 003546
1714 003546
1715 003546
1716
1717
1718
1719
1720 003546
1721 003546 052777 000001 175504
1722 003554
1723 003554 032777 000200 175476
1724 003562 001401

*****
*TEST 10 TEST THAT RCVRDONE IS CLEARED BY SETTING READER ENABLE
*****
TST10: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #10,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

IF #WRAP NOTSETIN $USWR OR CONSOL EQ #TRUE THEN
BIT #WRAP,$USWR
BEQ $41
CMP CONSOL,#TRUE
BNE $42
$41:
BR TST11 ;;EXIT THIS TEST
EXIT
ENDIF
$42:
BGNSUB
MOV #64,$LPERR
; OUTPUT A CHARACTER AND WAIT FOR XMITRDY TO SET.
; OUTPUT A CHARACTER
LET @TBUF :B- #0
; WAIT MAXIMUM OF 500 MSEC
; FOR RCVRDONE TO SET IN
; RCSR
CALL TIMER IN <#500,#RCVRDONE,RCSR,#SET>
MOV R5,-(SP)
MOV #SET,-(R5)
MOV RCSR,-(R5)
MOV #RCVRDONE,-(R5)
MOV #500,-(R5)
JSR PC,TIMER
MOV (SP)+,R5
; DID IT BECOME READY?
IF.ERROR THEN
;RCVRDONE DID NOT SET IN RCSR
ERRHRD 25
; SET IT BACK TO CONTINUE
ENDIF
$43:
ENDSUB
;NOW THAT IT IS SET LETS SEE IF SETTING
;READER ENABLE CLEARS RCVRDONE
LET @RCSR := @RCSR SET.BY #RDRRUN
IF #RCVRDONE SETIN @RCSR THEN
BIT #RCVRDONE,@RCSR
BEQ $44

```

CVDLAAO DLV11-J TEST MACY11 30G(1063) 17-NOV-78 09:39 PAGE 44  
CVDLAB.P11 17-NOV-78 09:38 T10 TEST THAT RCVRDONE IS CLEARED BY SETTING READER ENABLE SEQ 0043

1725

1726 003564

1727 003564 104026

1728

1729 003566

1730 003566

ERROR 26

\$44:

;RCVRDONE DID NOT CLEAR IN RCSR  
ERRHRD 26

; SET IT BACK TO CONTINUE  
ENDIF

```

1731
1732
1733
1734
1735 003566 000004
1736 003570 012737 000010 001160
1737 003576 012737 000011 001200
1738
1739 003604
1740 003604 032737 000020 001220
1741 003612 001404
1742 003614 023727 013604 000001
1743 003622 001001
1744 003624
1745 003624
1746 003624 000432
1747 003626
1748 003626
1749
1750 003626
1751 003626 012737 003634 001110
1752
1753
1754
1755 003634
1756 003634 105077 175426
1757
1758
1759
1760 003640
1761 003640 010546
1762 003642 012745 177777
1763 003646 013745 001260
1764 003652 012745 000200
1765 003656 012745 000500
1766 003662 004737 012446
1767 003666 012605
1768
1769 003670
1770 003670 103001
1771
1772 003672
1773 003672 104073
1774
1775 003674
1776 003674
1777 003674
1778
1779
1780
1781
1782
1783 003674
1784 003674 117700 175362
1785
1786 003700
  
```

```

*****
;TEST 11 TEST THAT RCVRDONE IS CLEARED BY READING RBUF
*****
TST11: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #11,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

IF #WRAP NOTSETIN $USWR OR CONSOL EQ #TRUE THEN
  BIT #WRAP,$USWR
  BEQ $45
  CMP CONSOL,#TRUE
  BNE $46
$45:
  BR TST12 ;;EXIT THIS TEST
  EXIT
  ENDIF
$46:
  BGNSUB
  MOV #64,$SLPERR
  ; OUTPUT A CHARACTER AND WAIT FOR RCVRDONE TO SET.
  ; OUTPUT A CHARACTER
  LET @TBUF :B- #0
  ; WAIT MAXIMUM OF 500 MSEC
  ; FOR RCVRDONE TO SET IN
  ; RCSR
  CALL TIMER IN <#500,#RCVRDONE,RCSR,#SET>
  MOV R5,-(SP)
  MOV #SET,-(R5)
  MOV RCSR,-(R5)
  MOV #RCVRDONE,-(R5)
  MOV #500,-(R5)
  JSR PC,TIMER
  MOV (SP)+,R5
  ; DID IT BECOME READY?
  IF .ERROR THEN
    ;RCVRDONE DID NOT SET IN RCSR
    ERRHRD 73
    ; SET IT BACK TO CONTINUE
    ENDIF
  $47:
  ENDSUB
  ; NOW THAT IT IS SET LETS SEE IF READING THE
  ; BUFFER CLEARS RCVRDONE.
  ; READ BUFFER
  LET R0 :B= @RBUF
  IF #RCVRDONE SETIN @RCSR THEN
  
```

CVDLAAO DLV11-J TEST MACY11 30G(1063) 17-NOV-78 09:39 PAGE 46  
CVDLAB.P11 17-NOV-78 09:38 T11 TEST THAT RCVRDONE IS CLEARED BY READING RBUF

SEQ 0045

1787 003700 032777 000200 175352 BIT #RCVRDONE, @RCSR  
1788 003706 001401 BEQ \$50  
1789  
1790 003710  
1791 003710 '04074 ERROR 74  
1792  
1793 003712  
1794 003712 \$50:

;RCVRDONE DID NOT CLEAR IN RCSR  
ERRHRD 74

; SET IT BACK TO CONTINUE  
ENDIF

```

1795
1796
1797
1798
1799 003712 000004
1800 003714 012737 000010 001160
1801 003722 012737 000012 001200
1802
1803 003730
1804 003730 005737 013604
1805 003734 001007
1806 003736
1807 003736 032737 000020 001220
1808 003744 001401
1809
1810 003746
1811 003746 000401
1812 003750
1813 003750
1814 003750 000524
1815 003752
1816 003752
1817 003752
1818 003752 000401
1819 003754
1820 003754
1821 003754 000522
1822 003756
1823 003756
1824
1825
1826
1827 003756
1828 003756 012737 003764 001110
1829
1830
1831
1832
1833 003764
1834 003764 105077 175276
1835
1836 003770
1837 003770 010546
1838 003772 012745 177777
1839 003776 013745 001260
1840 004002 012745 000200
1841 004006 012745 000500
1842 004012 004737 012446
1843 004016 012605
1844
1845
1846 004020
1847 004020 103001
1848
1849 004022
1850 004022 104016

*****
*TEST 12 TEST THE OVERRUN & ERROR BITS - RBUF 14
*****
TST12: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #12,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

IF CONSOL EQ #0 THEN
    TST CONSOL
    BNE $51
    IF #WRAP SETIN $USWR THEN
        ;; NULL ---EXECUTE TEST
    ELSE
        BR $53
        $52:
        BR TST13 ;;EXIT THIS TEST
        EXIT
        ENDF
        $53:
        ELSE
        BR $54
        $51:
        BR TST13 ;;EXIT THIS TEST
        EXIT
        ENDF
        $54:

BGNSUB
MOV #64,$SLPERR
;OUTPUT 2 CHARACTERS
;THIS SHOULD AN CAUSE OVERRUN ERROR.
;OUTPUT 1 CHARACTER
LET @TBUF :B- #0
CALL TIMER IN <#500,#RCVRDONE,RCR,$SET>
MOV R5,-(SP)
MOV #SET,-(R5)
MOV RCSR,-(R5)
MOV #RCVRDONE,-(R5)
MOV #500,-(R5)
JSR PC,TIMER
MOV (SP)+,R5

;DID IT SET IN TIME?
IF .ERROR THEN
    BCC $55
    ;RCVRDONE DID NOT SET IN RCSR
    ERRHRD 16
    ERROR 16
  
```

```

1851 004024
1852 004024 $55:
1853
1854 ;OUTPUT 2ND CHARACTER
1855 004024 LET @TBUF :B= #0
1856 004024 105077 175236 CLR B @TBUF
1857 ;LET OVERRUN HAPPEN
1858 004030 WAITMS 300.
1859 004030 010546 MOV R5, -(SP)
1860 004032 012745 000454 MOV #300, -(R5)
1861 004036 004737 012620 JSR PC, WAIT
1862 004042 012605 MOV (SP)+, R5
1863
1864 ;READ BUFFER AND ERROR BITS
1865 004044 LET R4 :- @RBUF
1866 004044 017704 175212 MOV @RBUF, R4
1867
1868 ;IT DIDN'T SET
1869 004050 IF #ORERR NOTSET IN R4 THEN
1870 004050 032704 040000 BIT #ORERR, R4
1871 004054 001002 BNE $56
1872 ;ORERR DID NOT SET IN RBUF
1873 004056 ERRHRD 101
1874 004056 104101 ERROR 101
1875
1876 ;NO USE COMPOUNDING ERRORS
1877 004060 EXIT
1878 004060 000460 BR TST13 ;;EXIT THIS TEST
1879 004062 ENDIF
1880 004062 $56:
1881 004062 ENDSUB
1882
1883 ;NOW SEE IF ERROR BIT SET WITH OVERRUN ERROR:
1884 004062 BGNSUB
1885 004062 012737 004070 001110 MOV #64$, $LPERR
1886 004070 IF #ERROR NOTSET IN R4 THEN
1887 004070 032704 100000 BIT #ERROR, R4
1888 004074 001002 BNE $57
1889
1890 ;ERROR DID NOT SET IN RBUF
1891 004076 ERRHRD 102
1892 004076 104102 ERROR 102
1893
1894 ;--WHEN ORERR SET.
1895 ;GET OUT NOW.
1896 004100 EXIT
1897 004100 000450 BR TST13 ;;EXIT THIS TEST
1898 004102 ENDIF
1899 004102 $57:
1900 004102 ENDSUB
1901
1902 BGNSUB
1903 004102 012737 004110 001110 MOV #64$, $LPERR
1904 ;CHECK REAL RBUF TO SEE IF ORERR IS STILL SET.
1905
1906 004110 IF #ORERR NOTSET IN @RBUF THEN
  
```

```

1907 004110 032777 040000 175144      BIT      #ORERR,@RBUF
1908 004116 001002                      BNE      $60
1909
1910                                ;READING RBUF CLEARED ORERR.
1911 004120                                ERRHRD 103
1912 004120 104103                      ERROR    103
1913                                ;SKIP REST OF TEST
1914 004122                                EXIT
1915 004122 000437                      BR      TST13      ;;EXIT THIS TEST
1916 004124                                ENDIF
1917 004124                                $60:
1918 004124                                ENDSUB
1919                                BGNSUB
1920 004124                                MOV      #64,$LPERR
1921 004124 012737 004132 001110      MOV      #64,$LPERR
1922                                ;READING RBUF ABOVE SHOULD ENABLE ERROR TO BE CLEARED
1923                                ;BY NEXT TRANSFER.
1924                                ;NOW SEE IF THEY CLEAR WHEN ANOTHER CHAR. IS RECEIVED
1925
1926                                ;SEND A CHARACTER AROUND.
1927 004132                                LET @TBUF :B= #0
1928 004132 105077 175130      CLRB      @TBUF
1929 004136                                CALL TIMER IN <#500,#RCVRDONE,RCSR,#SET>
1930 004136 010546      MOV      R5,-(SP)
1931 004140 012745 177777      MOV      #SET,-(R5)
1932 004144 013745 001260      MOV      RCSR,-(R5)
1933 004150 012745 000200      MOV      #RCVRDONE,-(R5)
1934 004154 012745 000500      MOV      #500,-(R5)
1935 004160 004737 012446      JSR      PC,TIMER
1936 004164 012605      MOV      (SP)+,R5
1937
1938                                ;DID IT SET IN TIME?
1939 004166                                IF.ERROR THEN
1940 004166 103001                      BCC      $61
1941                                ;RCVRDONE DID NOT SET IN RCSR
1942 004170                                ERRHRD 20
1943 004170 104020                      ERROR    20
1944 004172                                ENDF
1945 004172                                $61:
1946
1947 004172                                IF #ORERR SET IN @RBUF THEN
1948 004172 032777 040000 175062      BIT      #ORERR,@RBUF
1949 004200 001402                      BEQ      $62
1950                                ;ORERR DID NOT CLEAR IN RBUF
1951 004202                                ERRHRD 104
1952 004202 104104                      ERROR    104
1953
1954                                ;-AFTER RECEIVING ANOTHER CHAR
1955                                ;SKIP AROUND REST
1956 004204                                EXIT
1957 004204 000406                      BR      TST13      ;;EXIT THIS TEST
1958 004206                                ENDF
1959 004206                                $62:
1960
1961 004206                                IF #ERROR SET IN @RBUF THEN
1962 004206 032777 100000 175046      BIT      #ERROR,@RBUF

```

K 4  
CVDLAAO DLV11-J TEST MACY11 30G(1063) 17-NOV-78 09:39 PAGE 50  
CVDLAB.P11 17-NOV-78 09:38 112 TEST THE OVERRUN & ERROR BITS - RBUF 14 SEQ 0049

|      |        |        |       |       |                              |
|------|--------|--------|-------|-------|------------------------------|
| 1963 | 004214 | 001401 | BEQ   | \$63  |                              |
| 1964 |        |        |       |       | :ERROR DID NOT CLEAR IN RBUF |
| 1965 | 004216 |        |       |       | ERRHRD 105                   |
| 1966 | 004216 | 104105 | ERROR | 105   |                              |
| 1967 |        |        |       |       |                              |
| 1968 | 004220 |        |       |       | ENDIF                        |
| 1969 | 004220 |        | \$63: |       |                              |
| 1970 | 004220 |        |       |       | ENDSUB                       |
| 1971 | 004220 |        |       |       | EXIT                         |
| 1972 | 004220 | 000400 | BR    | TST13 | ::EXIT THIS TEST             |

1973  
 1974  
 1975  
 1976  
 1977  
 1978  
 1979  
 1980  
 1981  
 1982  
 1983  
 1984  
 1985  
 1986  
 1987  
 1988  
 1989  
 1990  
 1991  
 1992  
 1993  
 1994  
 1995  
 1996  
 1997  
 1998  
 1999  
 2000  
 2001  
 2002  
 2003  
 2004  
 2005  
 2006  
 2007  
 2008  
 2009  
 2010  
 2011  
 2012  
 2013  
 2014  
 2015  
 2016  
 2017  
 2018  
 2019  
 2020  
 2021  
 2022  
 2023  
 2024  
 2025  
 2026  
 2027  
 2028

004222 000004  
 004224 012737 000010 001160  
 004232 012737 000013 001200  
 004240  
 004240 005037 013616  
 004244  
 004244 013703 001256  
 004250  
 004250 062703 000004  
 004254 012723 013610  
 004260 012713 000340  
 004264  
 004264 012737 004272 001110  
 004272  
 004272 010546  
 004274 012745 177777  
 004300 013745 001264  
 004304 012745 000200  
 004310 012745 000500  
 004314 004737 012446  
 004320 012605  
 004322  
 004322 042777 000100 174734  
 004330  
 004342  
 004342 052777 000100 174714  
 004350  
 004350 010546  
 004352 012745 177777  
 004356 012745 013616  
 004362 012745 000001  
 004366 012745 000500  
 004372 004737 012446  
 004376 012605

```

*****
:TEST 13      TRANSMITTER INTERRUPT LOGIC TEST
:              LOGICALLY THIS IS 4 SEPARATE TESTS
:              A) DOES TRANSMITTER INTERRUPT LOGIC WORK
:              B) AT PRIORITY OF 0
:              C) AND ONLY ONCE
:              D) BUT NOT WITH INTERRUPT ENABLE CLEAR
*****
TST'3:  SCOPE
        MOV    #10,$TIMES      ;;DO 10 ITERATIONS
        MOV    #13,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
                                ;;CLEAR 'INTERRUPT OCCURED' FLAG
                                LET INTFLAG : #0
                                ;GET VECTOR ADDRESS
                                LET R3 :- DLVEC
                                ;FOR THE TRANSMITTER
                                LET R3 := R3 + #4
                                ;SET VECTOR TO POINT TO TRANS.SRV AT PRI
                                BGNSUB
                                ;; MAKE SURE THAT TRANSMITTER READY IS SET
                                CALL TIMER IN <#500,#XMITRDY,TCSR,#SET>

        MOV    R5,-(SP)
        MOV    #SET,-(R5)
        MOV    TCSR,-(R5)
        MOV    #XMITRDY,-(R5)
        MOV    #500,-(R5)
        JSR    PC,TIMER
        MOV    (SP)+,R5

                                ;CLEAR INTERRUPT ENABLE
                                LET @TCSR :- @TCSR CLR.BY #XMITIE

                                ;SET IT TO 0

                                ;NOW SET I.E. BIT
                                LET @TCSR : @TCSR SET.BY #XMITIE

                                CALL TIMER IN <#500,#1,#INTFLAG,#SET>

        SETPRI #PRO

        MOV    R5,-(SP)
        MOV    #SET,-(R5)
        MOV    #INTFLAG,-(R5)
        MOV    #1,-(R5)
        MOV    #500,-(R5)
        JSR    PC,TIMER
        MOV    (SP)+,R5
  
```

```

2029
2030                                     ;DID IT SET IN TIME?
2031 004400                                     IF.ERROR THEN
2032 004400 103001                                BCC      $64
2033                                     ;INTERRUPT DID NOT OCCUR
2034 004402                                     ERRHRD 106
2035 004402 104106                                ERROR   106
2036 004404                                     ENDIF
2037 004404                                     $64:
2038
2039                                     ;LET POSSIBLE 2'ND INTERR OCCUR
2040 004404                                     WAITMS 500.
2041 004404 010546                                MOV      R5,-(SP)
2042 004406 012745 000764                        MOV      #500,-(R5)
2043 004412 004737 012620                        JSR      PC,WAIT
2044 004416 012605                        MOV      (SP)+,R5
2045
2046                                     ;DID EXACTLY 1 INTERRUPT OCCUR
2047 004420                                     IF INTFLAG GT #1 THEN
2048 004420 023727 013616 000001                CMP      INTFLAG,#1
2049 004426 003401                                BLE      $65
2050                                     ;TRANSMITTER INTERRUPTED TWICE
2051 004430                                     ERRHRD 107
2052 004430 104107                                ERROR   107
2053 004432                                     ENDIF
2054 004432                                     $65:
2055 004432
2056                                     ENDSUB
2057                                     ;INTERRUPT WITHOUT INTERRUPT ENABLE SET
2058 004432 012737 004440 001110                MOV      #64$,$LPERR
2059                                     BGNSUB
2060 004440                                     ;CLEAR INTERRUPT ENABLE
2061 004440 042777 000100 174616                BIC      #XMITIE,@TCSR
2062                                     LET @TCSR := @TCSR CLR.BY #XMITIE
2063 004446                                     ;CLEAR 'INTERRUPT OCCURED' FLAG
2064 004446 005037 013616                CLR      INTFLAG
2065                                     LET INTFLAG := #0
2066                                     ;NO INTERRUPTS SHOULD OCCUR, PSW STILL AT 0.
2067 004452                                     ;DARE IT TO HAPPEN
2068 004452 005077 174610                CLR      @TBUF
2069                                     LET @TBUF := #0
2070                                     ;SEE IF INT FLAG EVER SETS
2071 004456                                     CALL TIMER IN <#1000,#1,#INTFLAG,#SET>
2072 004460 012745 177777                MOV      R5,-(SP)
2073 004464 012745 013616                MOV      #SET,-(R5)
2074 004470 012745 000001                MOV      #INTFLAG,-(R5)
2075 004474 012745 001000                MOV      #1,-(R5)
2076 004500 004737 012446                MOV      #1000,-(R5)
2077 004504 012605                JSR      PC,TIMER
2078                                     MOV      (SP)+,R5
2079 004506 103401                                BCS      1$
2080                                     ;DID IT SET DURING TIMER?
2081 004510                                     ;BR IF NO
2082 004510 104110                                ;INTERR STILL OCCURED WITH IE DISABLED
2083 004512                                     ERRHRD 110
                                     $:
                                     ERROR   110

```

```

2084
2085
2086
2087
2088
2089
2090
2091 004512 000004
2092 004514 012737 000010 001160
2093 004522 012737 000014 001200
2094 004530
2095 004530 032737 000020 001220
2096 004536 001404
2097 004540 023727 013604 000001
2098 004546 001002
2099 004550
2100 004550 000137 005046
2101 004554
2102 004554
2103
2104
2105
2106 004554
2107 004554 010146
2108 004556 013701 001256
2109 004562 012721 013610
2110 004566 012711 000340
2111 004572 012601
2112
2113 004574
2114 004574 012737 004602 001110
2115 004602
2116 004602 005037 013616
2117
2118 004606
2119 004606 042777 000100 174444
2120
2121
2122 004614
2123
2124
2125 004626
2126 004626 105077 174434
2127
2128
2129
2130 004632
2131 004632 010546
2132 004634 012745 177777
2133 004640 013745 001264
2134 004644 012745 000200
2135 004650 012745 000500
2136 004654 004737 012446
2137 004660 012605
2138
2139 004662

```

\*\*\*\*\*  
 \*TEST 14 RECEIVER INTERRUPT LOGIC TEST  
 \* THIS TEST COVERS ALL OF THE RECEIVER  
 \* SIDE OF THE INTERRUPT LOGIC IN  
 \* CHARACTER MODE.  
 \*\*\*\*\*  
 TST14: SCOPE  
 MOV #10,\$TIMES ;:DO 10 ITERATIONS  
 MOV #14,\$TESTN ;:SET TEST NUMBER IN APT MAIL BOX  
 IF #WRAP NOTSET IN \$USWR OR CONSOL EQ #TRUE THEN  
 BIT #WRAP,\$USWR  
 BEQ \$66  
 CMP CONSOL,#TRUE  
 BNE \$67  
 \$66: JMP TST15 ; EXIT TEST  
 \$67: ; ENDIF  
 ;CLEAR INTERRUPT OCCURED FLAG  
 ;SET UP RECEIVER INTER.VECTOR  
 SETVEC DLVEC,#INTSRV,#PR7  
 MOV R1,-(SP)  
 MOV DLVEC,R1  
 MOV #INTSRV,(R1)+  
 MOV #PR7,(R1)  
 MOV (SP)+,R1  
 ;PRIORITY 0 AND MULTIPLE INTERRUPT TEST.-RCVRIE  
 BGNSUB  
 MOV #64,\$LPERR  
 LET INTFLAG :- #0  
 CLR INTFLAG  
 ;CLEAR INTERRUPTS  
 LET @RCSR :- @RCSR CLR.BY #RCVRIE  
 ;CHANGE PRIORITY  
 ;...TO 0  
 SETPRI #PRO  
 ;SEND A CHARACTER  
 LET @TBUF :B= #0  
 ;WAIT A MAXIMUM  
 ;OF 50 MSEC FOR  
 ;XMIT RDY TO SET IN TCSR  
 CALL TIMER IN <#500,#RCVRDONE,TCSR,#SET>  
 MOV R5,-(SP)  
 MOV #SET,-(R5)  
 MOV TCSR,-(R5)  
 MOV #RCVRDONE,-(R5)  
 MOV #500,-(R5)  
 JSR PC,TIMER  
 MOV (SP)+,R5  
 ;SET INTERRUPT ENABLE  
 LET @RCSR :- @RCSR SET.BY #RCVRIE

```

2140 004662 052777 000100 174370      JIS      #RCVRIE,@RCSR
2141                                     ;LET IT COME IN.
2142                                     CALL TIMER IN <#500,#1,#INTFLAG,#SET>
2143 004670
2144 004670 010546      MOV      R5,-(SP)
2145 004672 012745 177777      MOV      #SET,-(R5)
2146 004676 012745 013616      MOV      #INTFLAG,-(R5)
2147 004702 012745 000001      MOV      #1,-(R5)
2148 004706 012745 000500      MOV      #500,-(R5)
2149 004712 004737 012446      JSR      PC,TIMER
2150 004716 012605      MOV      (SP)+,R5
2151
2152                                     ;DID IT SET IN TIME?
2153 004720                                     IF.ERROR THEN
2154 004720 103001      BCC      $70
2155                                     ;INTERRUPT DID NOT OCCUR
2156 004722                                     ERRHRD 111
2157 004722 104111
2158 004724                                     ENDIF
2159 004724                                     ;LET POSSIBLE 2'ND INTERR OCCUR
2160                                     WAITMS 500
2161 004724
2162 004724 010546      MOV      R5,-(SP)
2163 004726 012745 000500      MOV      #500,-(R5)
2164 004732 004737 012620      JSR      PC,WAIT
2165 004736 012605      MOV      (SP)+,R5
2166
2167                                     ;EXACTLY 1 INTERRUPT?
2168 004740                                     IF INTFLAG GT #1 THEN
2169 004740 023727 013616 000001      CMP      INTFLAG,#1
2170 004746 003401      BLE      $71
2171                                     ;RECEIVER INTERRUPTED TWICE
2172 004750                                     ERRHRD 112
2173 004750 104112
2174 004752                                     ENDIF
2175 004752                                     ;SEND A CHARACTER
2176 004752                                     LET @TBUF :B= #0
2177                                     ENDSUB
2178
2179
2180
2181
2182
2183                                     ;INTERRUPT WITHOUT IE SET.
2184 004752                                     BGNSUB
2185 004752 012737 004760 001110      MOV      #64$,$LPERR
2186
2187                                     ;CLEAR INTERRUPT
2188 004760                                     LET @RCSR :- @RCSR CLR.BY #RCVRIE
2189 004760 042777 000100 174272      BIC      #RCVRIE,@RCSR
2190                                     ;CLEAR INTERRUPT FLAG
2191 004766                                     LET INTFLAG := #0
2192 004766 005037 013616      CLR      INTFLAG
2193
2194 004772                                     ; SEND A CHARACTER
2195 004772 105077 174270      CLRB      @TBUF

```

```

2196                                     ; DARE IT
2197                                     CALL TIMER IN <#500,#1,#INTFLAG,#CLR>
2198 004776                                     MOV     R5,-(SP)
2199 004776 010546                               MOV     #CLR,-(R5)
2200 005000 012745 000000                       MOV     #INTFLAG,-(R5)
2201 005004 012745 013616                       MOV     #1,-(R5)
2202 005010 012745 000001                       MOV     #500,-(R5)
2203 005014 012745 000500                       JSR     PC,TIMER
2204 005020 004737 012446                       MOV     (SP)+,R5
2205 005024 012605
2206                                     ;DID IT CLEAR IN TIME?
2207 005026                                     IF.ERROR THEN
2208 005026 103001                               BCC     $72
2209
2210                                     ;INTERR STILL OCCURED WITH IE DISABLED
2211 005030 104113                               ERRHRD 113
2212 005032                                     ENDIF
2213 005032                                     $72:
2214 005032                                     ENDSUB
2215 005032                               SETPRI #PR7 ;RAISE CPU PRIORITY
2216                                     ;CLEAR THE WORLD
2217 005044                                     BRESET
2218 005044 000005                               RESET
2219
2220

```

```

2221
2222
2223 *****
2224 :*TEST 15      TEST DATA  WRAP AROUND: FLAG MODE
2225 *****
2225 005046 000004 TST15: SCOPE
2226 005050 012737 000001 001160 MOV #1,$TIMES ;;DO 1 ITERATION
2227 005056 012737 000015 001200 MOV #15,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
2228 005064 032737 000020 001220 BIT #WRAP,$USWR IF #WRAP NOTSET IN $USWR OR CONSOL EQ #TRUE THEN
2229 005064 001404 BEQ $73
2230 005072 023727 013604 000001 CMP CONSOL,#TRUE
2231 005102 001001 BNE $74
2232 005104 $73:
2233
2234
2235 005104 BR TST16 ;;EXIT THIS TEST
2236 005104 000506 EXIT ;CAN'T TEST WITHOUT A WRAP
2237 005106 ENDIF
2238 005106 $74:
2239 005106 BRESET
2240 005106 000005 RESET
2241
2242
2243 005110 CLR R2 ;BINARY COUNT PATTERN
2244 005112 000401 BR $75 INCR R2 FROM #0 TO #377 BY #1
2245 005114 $76:
2246 005114 005202 INC R2
2247 005116 $75:
2248 005116 020227 000377 CMP R2,#377
2249 005122 003077 BGT $77
2250
2251
2252
2253
2254 005124
2255 005124 010546 MOV R5,-(SP)
2256 005126 012745 177777 MOV #SET,-(R5)
2257 005132 013745 001264 MOV TCSR,-(R5)
2258 005136 012745 000200 MOV #XMITRDY,-(R5)
2259 005142 012745 000500 MOV #500,-(R5)
2260 005146 004737 012446 JSR PC,TIMER
2261 005152 012605 MOV (SP)+,R5
2262 005154
2263 005154 103002 BCC $100 IF.ERROR THEN
2264
2265 005156
2266 005156 104123 ERROR 123 ; TRANSMITTER NEVER BECAME READY
2267 005160 EXIT ERRHRD 123
2268 005160 000460 BR TST16 ;;EXIT THIS TEST
2269 005162 $100:
2270 005162
2271
2272
2273 005162
2274 005162 110277 174100 MOVB R2,@TBUF ;START IT ON ITS WAY
2275
2276 005166
  
```

CALL TIMER IN <#500,#RCVRDONE,RCR,#SET>  
 ; NOW WAIT FOR RECIEVER DONE

```

2277 005166 010546      MOV      R5,-(SP)
2278 005170 012745 177777  MOV      #SET,-(R5)
2279 005174 013745 001260  MOV      RCSR,-(R5)
2280 005200 012745 000200  MOV      #RCVRDONE,-(R5)
2281 005204 012745 000500  MOV      #500,-(R5)
2282 005210 004737 012446  JSR      PC,TIMER
2283 005214 012605      MOV      (SP)+,R5
2284 005216                      IF.ERROR THEN
2285 005216 103002      BCC      $101
2286 005220                      ERRHRD 124
2287 005220 104124      ERROR    124
2288                      ; RECIEVER NEVER BECAME READY
2289 005222                      EXIT
2290 005222 000437      BR       TST16      ;;EXIT THIS TEST
2291 005224                      ENDIF
2292 005224      $101:
2293
2294                      ;RETRIEVE
2295 005224                      LET R3 := @RBUF
2296 005224 017703 174032      MOV      @RBUF,R3
2297                      ;CHECK FOR ERROR DURING TRANSFER
2298 005230                      IF #ERROR SETIN R3 THEN
2299 005230 032703 100000      BIT      #ERROR,R3
2300 005234 001401      BEQ      $102
2301                      ;ERROR BIT SET IN HIGH BYTE OF RBUF
2302 005236                      ERRHRD 200
2303 005236 104200      ERROR    200
2304 005240                      ENDIF
2305 005240      $102:
2306
2307                      ;COMPARE DATA
2308                      IF R3 NE R2 THEN
2309 005240                      CMP      R3,R2
2310 005240 020302      BEQ      $103
2311 005242 001415
2312 005244                      IF #BIT0 NOTSETIN $USWR THEN
2313 005244 032737 000001 001220      BIT      #BIT0,$USWR
2314 005252 001006      BNE      $104
2315 005254                      IF R2 LT #200 THEN
2316 005254 020227 000200      CMP      R2,#200
2317 005260 002002      BGE      $105
2318                      ;DATA COMPARE ERR IN 7 BIT WORD
2319 005262                      ERRHRD 117
2320 005262 104117      ERROR    117
2321 005264                      EXIT
2322 005264 000416      BR       TST16      ;;EXIT THIS TEST
2323 005266                      ENDIF
2324 005266      $105:
2325 005266                      ELSE
2326 005266 000402      BR       $106
2327 005270      $104:
2328                      ;DATA COMP ERR IN 8 BIT WORD
2329 005270                      ERRHRD 17
2330 005270 104017      ERROR    17
2331 005272                      EXIT
2332 005272 000413      BR       TST16      ;;EXIT THIS TEST

```

```

2333 005274
2334 005274
2335 005274
2336 005274 000411
2337 005276
2338 005276 032737 000001 001220
2339 005304 001005
2340 005306 020227 000177
2341 005312 003407
2342
2343
2344 005314
2345 005314 104022
2346 005316
2347 005316 000401
2348 005320
2349 005320
2350 005320
2351 005320
2352 005320 000675
2353 005322

$106:
$103:
    BR      $107
    BIT     #BIT0,$USWR
    BNE     $110
    CMP     R2,#177
    BLE     $110

    ;GETTING 8 BITS ON 7 BIT XMIT
    ;MAKE SURE $USWR SETUP CORRECTLY.
    ERRHRD 22

    ERROR   22

    BR      TST16

    $110:
    $107:

    BR      $76

    $77:

    ENDIF
    ELSE IF #BIT0 NOTSET IN $USWR AND R2 GT #177 THEN
    EXIT
    ;;EXIT THIS TEST
    ENDIF

    ENDINC ;R2
  
```

```

2354
2355
2356
2357
2358 005322 000004
2359 005324 012737 000001 001160
2360 005332 012737 000016 001200
2361 005340
2362 005340 032737 000020 001220
2363 005346 001404
2364 005350 023727 013604 000001
2365 005356 001002
2366 005360
2367 005360 000137 005772
2368 005364
2369 005364
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387 005364
2388
2389 005376
2390 005376 013701 001256
2391
2392 005402
2393 005402 012721 005664
2394 005406
2395 005406 012721 000340
2396
2397
2398 005412
2399 005412 012721 005626
2400 005416
2401 005416 012711 000340
2402
2403
2404 005422
2405 005422 005037 005622
2406 005426
2407 005426 005037 005662
2408 005432
2409 005432 032737 000001 001220

*****
*TEST 16 TEST DATA WRAP AROUND: INTERRUPT MODE
*****
TST16: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
MOV #16,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
IF #WRAP NOTSET IN $USWR OR CONSOL EQ #TRUE THEN
BIT #WRAP,$USWR
BEQ $111
CMP CONSOL,#TRUE
BNE $112
$111: JMP TST17 ;EXIT TEST
ENDIF
$112:

: THIS TEST WILL RUN BOTH TRANSMITTER AND
: RECIEVER AT FULL SPEED TESTING
: THE ABILITY OF THE MODULE
: TO HANDLE INTERRUPTS FROM BOTH SIDES AT ONCE.
:
: DOUBLE BUFFERING IS NOT FULLY TESTED BECAUSE OF
: APT CONSIDERATIONS. I.E. 'BREAK' FROM APT
: CAUSES OVERRUN ERRORS. THEREFORE TRANSMIT INTR IS
: ENABLED ONLY AFTER THE RECVR HAS OBTAINED THE LAST WORD
:
: THIS TEST WILL TRANSFER A MAXIMUM OF 400(8)
: CHARACTERS THROUGH THE MODULE, BUT IF AN ERROR
: IS DETECTED BY THE TEST A PREMATURE SHUTDOWN OCCURS.
:
: CHANGE PRIORITY
: ...TO 0
SETPRI #PRO
: GET VECTOR ADDRESS
LET R1 := DLVEC
: RCVR VECTOR
LET (R1)+ := #REC
LET (R1)+ := #PR7
: POINT TO TRANSMITTER VECTOR
: AND SET IT UP ALSO
LET (R1)+ := #TRAN
LET (R1) : #PR7
: CLEAR ERROR COUNTER
LET ERRCNT := #0
LET DATA : #0 ;XMIT DATA
IF #BIT0 NOTSET IN $USWR THEN

```

|      |        |        |        |        |               |                                      |
|------|--------|--------|--------|--------|---------------|--------------------------------------|
| 2410 | 005440 | 00'004 |        | BNE    | \$113         | LET NUMBER := #200                   |
| 2411 | 005442 |        |        |        |               |                                      |
| 2412 | 005442 | 012737 | 000200 | MOV    | #200,NUMBER   | ELSE                                 |
| 2413 | 005450 |        |        |        |               |                                      |
| 2414 | 005450 | 000403 |        | BR     | \$114         |                                      |
| 2415 | 005452 |        |        | \$113: |               | LET NUMBER := #400                   |
| 2416 | 005452 |        |        |        |               |                                      |
| 2417 | 005452 | 012737 | 000400 | MOV    | #400,NUMBER   | ENDIF                                |
| 2418 | 005460 |        |        |        |               |                                      |
| 2419 | 005460 |        |        | \$114: |               |                                      |
| 2420 |        |        |        |        |               |                                      |
| 2421 | 005460 |        |        |        |               | BRESET ;SET UP ALL REGISTERS         |
| 2422 | 005460 | 000005 |        | RESET  |               |                                      |
| 2423 |        |        |        |        |               |                                      |
| 2424 |        |        |        |        |               | ;SET I.E. IN TRANSMITTER             |
| 2425 | 005462 |        |        |        |               | LET @TCSR := @TCSR SET.BY #XMITIE    |
| 2426 | 005462 | 052777 | 000100 | BIS    | #XMITIE,@TCSR |                                      |
| 2427 |        |        |        |        |               | ;AND RECEIVER                        |
| 2428 | 005470 |        |        |        |               | LET @RCR := @RCR SET.BY #RCVRIE      |
| 2429 | 005470 | 052777 | 000100 | BIS    | #RCVRIE,@RCR  |                                      |
| 2430 |        |        |        |        |               |                                      |
| 2431 |        |        |        |        |               |                                      |
| 2432 |        |        |        |        |               | ;NOW WE WAIT                         |
| 2433 | 005476 |        |        |        |               | REPEAT                               |
| 2434 | 005476 |        |        | \$115: |               |                                      |
| 2435 | 005476 |        |        |        |               | UNTIL DATA EQ NUMBER OR ERRCNT GT #0 |
| 2436 | 005476 | 023737 | 005662 | CMP    | DATA,NUMBER   |                                      |
| 2437 | 005504 | 001403 |        | BEQ    | \$116         |                                      |
| 2438 | 005506 | 005737 | 005622 | TST    | ERRCNT        |                                      |
| 2439 | 005512 | 003771 |        | BLE    | \$115         |                                      |
| 2440 | 005514 |        |        | \$116: |               |                                      |
| 2441 |        |        |        |        |               |                                      |
| 2442 |        |        |        |        |               | ;NOW LETS CHECK.                     |
| 2443 |        |        |        |        |               | ;TURN OFF ALL INTR ENABLE            |
| 2444 | 005514 |        |        |        |               | LET @TCSR := @TCSR CLR.BY #XMITIE    |
| 2445 | 005514 | 042777 | 000100 | BIC    | #XMITIE,@TCSR |                                      |
| 2446 | 005522 |        |        |        |               | LET @RCR := @RCR CLR.BY #RCVRIE      |
| 2447 | 005522 | 042777 | 000100 | BIC    | #RCVRIE,@RCR  |                                      |
| 2448 | 005530 |        |        |        |               | IF ERRCNT NE #0 THEN                 |
| 2449 | 005530 | 005737 | 005622 | TST    | ERRCNT        |                                      |
| 2450 | 005534 | 001431 |        | BEQ    | \$117         |                                      |
| 2451 | 005536 |        |        |        |               | IF #ERROR SETIN RHL D THEN           |
| 2452 | 005536 | 032737 | 100000 | BIT    | #ERROR,RHLD   |                                      |
| 2453 | 005544 | 001424 |        | BEQ    | \$120         |                                      |
| 2454 | 005546 |        |        |        |               | IF #ORERR SETIN RHL D THEN           |
| 2455 | 005546 | 032737 | 040000 | BIT    | #ORERR,RHLD   |                                      |
| 2456 | 005554 | 001402 |        | BEQ    | \$121         |                                      |
| 2457 |        |        |        |        |               | ;OVERRUN ERROR                       |
| 2458 | 005556 |        |        |        |               | ERRHRD 220                           |
| 2459 | 005556 | 104220 |        | ERROR  | 220           |                                      |
| 2460 | 005560 |        |        |        |               | ELSE IF #FRERR SETIN RHL D THEN      |
| 2461 | 005560 | 000415 |        | BR     | \$122         |                                      |
| 2462 | 005562 |        |        | \$121: |               |                                      |
| 2463 | 005562 | 032737 | 020000 | BIT    | #FRERR,RHLD   |                                      |
| 2464 | 005570 | 001402 |        | BEQ    | \$123         |                                      |
| 2465 |        |        |        |        |               | ;FRAMING ERROR                       |

```

2466 005572
2467 005572 104221          ERROR 221          ERRHRD 221
2468 005574
2469 005574 000407          BR $124          ELSE IF #PERR SET IN RHLD THEN
2470 005576
2471 005576 032737 010000 005770 $123:      BIT #PERR,RHLD
2472 005604 001402          BEQ $125
2473
2474 005606
2475 005606 104222          ERROR 222          ;PARITY ERROR
2476 005610
2477 005610 000401          BR $126          ERRHRD 222
2478 005612          $125:      ELSE
2479
2480 005612
2481 005612 104024          ERROR 24          ;UNKNOWN ERROR
2482 005614
2483 005614          $126:      ERRHRD 24
2484 005614          $124:      ENDIF
2485 005614          $122:
2486 005614
2487 005614 000401          BR $127          ELSE
2488 005616          $120:
2489
2490 005616
2491 005616 104120          ERROR 120          ;DATA COMPARE ERROR
2492 005620
2493 005620          $127:      ERRHRD 120
2494 005620
2495 005620          $117:      ENDIF
2496 005620
2497 005620 000464          BR TST17          EXIT ;SKIP OVER SUPPORT ROUTINES & STORAGE
2498
2499
2500 005622 000000          ERRCNT: 0
2501 005624 000000          NUMBER: 0

```

```

2502
2503 ;*****
2504 ;TRANSMIT INTERRUPT HANDLER
2505 ;*****
2506 BGNSRV TRAN
2507 TRAN:
2508 IF DATA NE NUMBER AND ERRCNT EQ #0 THEN
2509 CMP DATA,NUMBER
2510 BEQ $130
2511 TST ERRCNT
2512 BNE $130
2513 ;SHIP OUT WORD
2514 LET @TBUF :- DATA
2515 MOV DATA,@TBUF
2516 ENDIF
2517 $130:
2518 ;STOP INTERR, NOT EXER DOUBL BUFFER
2519 LET @TCSR :- @TCSR CLR.BY #XMITIE
2520 BIC #XMITIE,@TCSR
2521 ENDSRV
2522 RTI
2523
2524 DATA: 0
2525
2526 ;*****
2527 ;RECEIVER INTERRUPT HANDLER
2528 ;*****
2529 BGNSRV REC
2530 REC:
2531 ;GET CHAR
2532 LET RHLA :- @RBUF
2533 MOV @RBUF,RHLA
2534 ;CHECK ERROR
2535 IF #ERROR SETIN RHLA OR RHLA NE DATA THEN
2536 BIT #ERROR,RHLA
2537 BNE $131
2538 CMP RHLA,DATA
2539 BEQ $132
2540 $131:
2541 ;STOP ALL INTERR PROC & GET OUT
2542 LET DATA := NUMBER
2543 MOV NUMBER,DATA
2544 LET @RCSR := @RCSR CLR.BY #RCVRIE
2545 BIC #RCVRIE,@RCSR
2546 LET ERRCNT := ERRCNT + #1
2547 INC ERRCNT
2548 ELSE
2549 BR $133
2550 $132:
2551 LET DATA := DATA + #1
2552 INC DATA
2553 IF DATA EQ NUMBER THEN
2554 CMP DATA,NUMBER
2555 BNE $134
2556 LET @RCSR : @RCSR CLR.BY #RCVRIE
2557 BIC #RCVRIE,@RCSR

```

K 5

(VDLAAO DLV11-J TEST MACY11 30G(1063) 17-NOV-78 09:39 PAGE 63  
 (VDLAB.P11 17-NOV-78 09:38 T16 TEST DATA WRAP AROUND: INTERRUPT MODE

SEQ 0062

```

2558 005756                                     ELSE
2559 005756 000403                               BR      $135
2560 005760                                     $134:
2561                                     ;ALLOW NEXT XMIT INTERR
2562 005760                                     LET @TCSR := @TCSR SET.BY #XMITIE
2563 005760 052777 000100 173276               BIS      #XMITIE,@TCSR
2564 005766                                     ENDIF
2565 005766                                     $135:
2566 005766                                     ENDIF
2567 005766                                     $133:
2568 005766                                     ENDSRV
2569 005766 000002                               RTI
2570
2571 005770 000000                               RHL D: 0
2572
  
```

```

2573
2574
2575
2576
2577
2578
2579
2580
2581 005772 000004
2582 005774 012737 000010 001160
2583 006002 012737 000017 001200
2584
2585
2586
2587 006010
2588 006010 032737 000020 001220
2589 006016 001404
2590 006020 032737 000010 001220
2591 006026 001003
2592 006030
2593 006030 000137 006502
2594 006034
2595 006034 000406
2596 006036
2597 006036
2598 006036 023727 013604 000000
2599 006044 001002
2600
2601 006046 000137 006502
2602 006052
2603 006052
2604 006052
2605 006052
2606 006052
2607 006052 012737 006060 001110
2608
2609 006060
2610 006060 005037 006500
2611
2612 006064
2613 006064 052777 000001 173172
2614
2615 006072
2616 006072 012777 000125 173166
2617
2618 006100
2619 006100 010546
2620 006102 012745 177777
2621 006106 013745 001260
2622 006112 012745 000200
2623 006116 012745 000500
2624 006122 004737 012446
2625 006126 012605
2626 006130
2627 006130 103001
2628

*****
:TEST 17      TEST BREAK DETECTION LOGIC
:              TRANSMIT KNOWN CHAR WITH BREAK SET
:              AND COMPARE RECEIVED WITH 0.
:              FRAMING ERROR WILL ALSO BE CHECKED
:              IF ERROR BITS ARE ENABLED.
*****
TST17: SCOPE
MOV      #10,$TIMES      ;;DO 10 ITERATIONS
MOV      #17,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
                        ;;DONT DO THIS TEST IF 'BREAK' GENERATION
                        ;;ENABLED, ELSE WILL HALT TO CONSOLE ODT.
                        ;;DO IF BREAK 'DETECTION' IS ENABLED.
                        IF #WRAP NOTSETIN $USWR OR #BRK NOTSETIN $USWR THEN

BIT      #WRAP,$USWR
BEQ      $136
BIT      #BRK,$USWR
BNE      $137

$136:      JMP      TST20      ;EXIT TEST
                        ELSE
BR        $140
$137:
                        IF CONSOL EQ #TRJE THEN
CMP      CONSOL,#TRUE
BNE      $141
                        ;CAN'T TEST CONSOLE
                        ;EXIT TEST
                        ENDIF
JMP      TST20
$141:
ENDIF
$140:
BGNSUB
MOV      #64,$LPERR
LET ERRCHK := #0      , CLEAR ERROR WORD
CLR      ERRCHK
                        ;SET BREAK BIT
LET @TCSR := @TCSR SET.BY #BREAK
                        ;NON-ZERO CHAR. '*'
LET @TBUF := #125
                        ;WAIT FOR DONE
CALL TIMER IN <#500,#RCVRDONE,RCSR,#SET>

MOV      R5,-(SP)
MOV      #SET,-(R5)
MOV      RCSR,-(R5)
MOV      #RCVRDONE,-(R5)
MOV      #500,-(R5)
JSR      PC,TIMER
MOV      (SP)+,R5
IF.ERROR THEN
                        ; RECIEVER DONE DID NOT SE'

```

|      |        |        |        |        |        |       |                 |  |
|------|--------|--------|--------|--------|--------|-------|-----------------|--|
| 2629 | 006132 |        |        |        |        |       |                 |  |
| 2630 | 006132 | 104115 |        |        |        | ERROR | 115             |  |
| 2631 | 006134 |        |        |        |        |       |                 |  |
| 2632 | 006134 |        |        |        | \$142: |       |                 |  |
| 2633 | 006134 |        |        |        |        |       |                 |  |
| 2634 | 006134 | 017700 | 173122 |        |        | MOV   | @RBUF,R0        |  |
| 2635 | 006140 |        |        |        |        |       |                 |  |
| 2636 | 006140 | 105700 |        |        |        | TSTB  | R0              |  |
| 2637 | 006142 | 001403 |        |        |        | BEQ   | \$143           |  |
| 2638 |        |        |        |        |        |       |                 |  |
| 2639 | 006144 |        |        |        |        |       |                 |  |
| 2640 | 006144 | 052737 | 000001 | 006500 |        | BIS   | #BIT0,ERRCHK    |  |
| 2641 | 006152 |        |        |        |        |       |                 |  |
| 2642 | 006152 |        |        |        | \$143: |       |                 |  |
| 2643 | 006152 |        |        |        |        |       |                 |  |
| 2644 | 006152 | 032700 | 020000 |        |        | BIT   | #FRERR,R0       |  |
| 2645 | 006156 | 001003 |        |        |        | BNE   | \$144           |  |
| 2646 | 006160 |        |        |        |        |       |                 |  |
| 2647 | 006160 | 052737 | 000002 | 006500 |        | BIS   | #BIT1,ERRCHK    |  |
| 2648 | 006166 |        |        |        |        |       |                 |  |
| 2649 | 006166 |        |        |        | \$144: |       |                 |  |
| 2650 | 006166 |        |        |        |        |       |                 |  |
| 2651 | 006166 | 032737 | 000002 | 001220 |        | BIT   | #PARITY,\$USWR  |  |
| 2652 | 006174 | 001421 |        |        |        | BEQ   | \$145           |  |
| 2653 |        |        |        |        |        |       |                 |  |
| 2654 | 006176 |        |        |        |        |       |                 |  |
| 2655 | 006176 | 032737 | 000004 | 001220 |        | BIT   | #EVENODD,\$USWR |  |
| 2656 | 006204 | 001007 |        |        |        | BNE   | \$146           |  |
| 2657 |        |        |        |        |        |       |                 |  |
| 2658 | 006206 |        |        |        |        |       |                 |  |
| 2659 | 006206 | 032700 | 010000 |        |        | BIT   | #PERR,R0        |  |
| 2660 | 006212 | 001003 |        |        |        | BNE   | \$147           |  |
| 2661 |        |        |        |        |        |       |                 |  |
| 2662 | 006214 |        |        |        |        |       |                 |  |
| 2663 | 006214 | 052737 | 000004 | 006500 |        | BIS   | #BIT2,ERRCHK    |  |
| 2664 | 006222 |        |        |        |        |       |                 |  |
| 2665 | 006222 |        |        |        | \$147: |       |                 |  |
| 2666 | 006222 |        |        |        |        |       |                 |  |
| 2667 | 006222 | 000406 |        |        |        | BR    | \$150           |  |
| 2668 | 006224 |        |        |        | \$146: |       |                 |  |
| 2669 | 006224 |        |        |        |        |       |                 |  |
| 2670 | 006224 | 032700 | 010000 |        |        | BIT   | #PERR,R0        |  |
| 2671 | 006230 | 001403 |        |        |        | BEQ   | \$151           |  |
| 2672 | 006232 |        |        |        |        |       |                 |  |
| 2673 | 006232 | 052737 | 000010 | 005622 |        | BIS   | #BIT3,ERRCNT    |  |
| 2674 | 006240 |        |        |        |        |       |                 |  |
| 2675 | 006240 |        |        |        | \$151: |       |                 |  |
| 2676 | 006240 |        |        |        |        |       |                 |  |
| 2677 | 006240 |        |        |        | \$150: |       |                 |  |
| 2678 | 006240 |        |        |        |        |       |                 |  |
| 2679 | 006240 |        |        |        | \$145: |       |                 |  |
| 2680 |        |        |        |        |        |       |                 |  |
| 2681 | 006240 |        |        |        |        |       |                 |  |
| 2682 | 006240 | 000005 |        |        |        | RESET |                 |  |
| 2683 | 006242 | 032777 | 170000 | 173012 |        | BIT   | #70000,@RBUF    |  |
| 2684 | 006250 | 001401 |        |        |        | BEQ   | 15              |  |

```

ERRHRD 115
ENDIF
LET R0 := @RBUF
IFB R0 NE #0 THEN
    ; BREAK DID NOT EQUAL 0
    LET ERRCHK := ERRCHK SET.BY #BIT0
ENDIF
IF #FRERR NOTSETIN R0 THEN
    LET ERRCHK : ERRCHK SET.BY #BIT1
ENDIF
IF #PARITY SETIN $USWR THEN
    ;ODD PARITY ENABLED
    IF #EVENODD NOTSETIN $USWR THEN
        ;BREAK SHOULD GENERATE A PARITY ERRO
        IF #PERR NOTSETIN R0 THEN
            ;NO PAR ERROR WHEN THERE SHOULD
            LET ERRCHK : ERRCHK SET.BY #BIT2
        ENDIF
    ELSE
        IF #PERR SETIN R0 THEN
            LET ERRCNT := ERRCNT SET.BY #BIT3
        ENDIF
    ENDIF
ENDIF
BRESET ;CLEAN UP

```

```

2685 006252 104033          ERROR 33          ;RESET DID NOT CLEAR ERROR,FR ERR,OR PERR IN RBUF
2686 006254          1$:
2687
2688 006254
2689 006254 032737 000001 006500      BIT      #BIT0,ERRCHK
2690 006262 001401          BEQ      $152
2691 006264
2692 006264 104121          ERROR 121
2693 006266
2694 006266          $152:
2695 006266
2696 006266 032737 000002 006500      BIT      #BIT1,ERRCHK
2697 006274 001401          BEQ      $153
2698 006276
2699 006276 104122          ERROR 122
2700 006300
2701 006300          $153:
2702 006300
2703 006300 032737 000004 006500      BIT      #BIT2,ERRCHK
2704 006306 001401          BEQ      $154
2705 006310
2706 006310 104235          ERROR 235
2707
2708
2709 006312
2710 006312          $154:
2711 006312
2712 006312 032737 000010 006500      BIT      #BIT3,ERRCHK
2713 006320 001401          BEQ      $155
2714 006322
2715 006322 104236          ERROR 236
2716
2717
2718
2719 006324
2720 006324          $155:
2721 006324
2722 006324
2723 006324 012737 006332 001110      MOV      #64$,$LPERR
2724
2725 006332
2726 006332 052777 000001 172724      BIS      #BREAK,@TCSR
2727
2728 006340
2729 006340 010546          MOV      R5,-(SP)
2730 006342 012745 177777          MOV      #SET,-(R5)
2731 006346 013745 001264          MOV      TCSR,-(R5)
2732 006352 012745 000001          MOV      #BREAK,-(R5)
2733 006356 012745 000500          MOV      #500,-(R5)
2734 006362 004737 012446          JSR      PC,TIMER
2735 006366 012605          MOV      (SP)+,R5
2736
2737
2738 006370
2739 006370 103001          BCC      $+56
2740

```

IF #BIT0 SETIN ERRCHK THEN

ERRHRD 121 ;BREAK ERROR

ENDIF

IF #BIT1 SETIN ERRCHK THEN

ERRHRD 122 ; FRAMING ERROR

ENDIF

IF #BIT2 SETIN ERRCHK THEN

ERRHRD 235

;NO PARITY ERROR WHEN  
;THERE SHOULD BE

ENDIF

IF #BIT3 SETIN ERRCHK THEN

ERRHRD 236

;PARITY ERROR SHOULD NOT HAVE  
;OCCURED WITH EVEN PARITY  
;ENABLED AND BREAK SET

ENDIF

ENDSUB  
BGNSUB

;SET BREAK BIT  
LET @TCSR := @TCSR SET.BY #BREAK

CALL TIMER IN <#500,#BREAK,TCSR,#SET>

;DID IT SET?  
IF.ERROR THEN

;BREAK DID NOT SET

ERRHRD 21

```

2741 006372
2742 006372 104021
2743 006374
2744 006374
2745
2746
2747 006374
2748 006374 042777 000001 172662
2749 006402
2750 006402 010546
2751 006404 012745 000144
2752 006410 004737 012620
2753 006414 012605
2754
2755
2756
2757 006416
2758 006416 017700 172640
2759
2760 006422
2761 006422 012777 000125 172636
2762
2763 006430
2764 006430 010546
2765 006432 012745 177777
2766 006436 013745 001260
2767 006442 012745 000200
2768 006446 012745 000500
2769 006452 004737 012446
2770 006456 012605
2771 006460
2772 006460 103001
2773
2774 006462
2775 006462 104230
2776 006464
2777 006464
2778
2779
2780 006464
2781 006464 127727 172572 000125
2782 006472 001401
2783
2784 006474
2785 006474 104231
2786 006476
2787 006476
2788 006476
2789 006476
2790 006476 000401
2791 006500 000000

          $156:
          BIC      #BREAK,@TCSR
          MOV      R5,-(SP)
          MOV      #100,-(R5)
          JSR      PC,WAIT
          MOV      (SP)+,R5

          :READ RBUF TO CLEAR ERRORS & REC DONE
          LET R0 := @RBUF

          :SEND CHAR
          LET @TBUF := #125

          :WAIT FOR DONE BIT
          CALL TIMER IN <#500,#RCVRDONE,RCSR,#SET>

          IF .ERROR THEN
              :RECEIVER NEVER CAME READY
              ERRHRD 230
          ENDIF

          :WAS CHAR AFTER BREAK RECEIVED
          IFB @RBUF NE #125 THEN
              :CHAR AFTER BREAK NOT RECEIVED CORRECTLY
              ERRHRD 231
          ENDIF

          $157:
          CMPB     @RBUF,#125
          BEQ      $160

          ERROR    231

          $160:
          BR       TST20
          ERRCHK: .WORD 0
  
```

ENDIF

```

: CLEAR BREAK BIT
LET @TCSR := @TCSR CLR.BY #BREAK

WAITMS 100.
  
```

```

: READ RBUF TO CLEAR ERRORS & REC DONE
LET R0 := @RBUF
  
```

```

: SEND CHAR
LET @TBUF := #125
  
```

```

: WAIT FOR DONE BIT
CALL TIMER IN <#500,#RCVRDONE,RCSR,#SET>
  
```

IF .ERROR THEN

```

: RECEIVER NEVER CAME READY
ERRHRD 230
  
```

ENDIF

```

: WAS CHAR AFTER BREAK RECEIVED
IFB @RBUF NE #125 THEN
  
```

```

: CHAR AFTER BREAK NOT RECEIVED CORRECTLY
ERRHRD 231
  
```

ENDIF

```

ENDSUB
EXIT
::EXIT THIS TEST
  
```

```

2792
2793
2794
2795
2796 006502 000004
2797 006504 012737 000001 001160
2798 006512
2799 006512 032777 010000 172420
2800 006520 001513
2801 006522
2802 006522 023727 014166 000001
2803 006530 001021
2804 006532
2805 006572
2806 006572 000425
2807 006574
2808 006574
2809 006574 005737 014170
2810 006600 001022
2811 006602
2812 006642
2813 006642 005237 014170
2814 006646
2815 006646
2816 006646
2817 006646
2818 006646
2819 006662
2820 006670
2821 006710
2822 006716
2823 006736
2824 006744 104401 001171
2825 006750
2826 006750
2827 006750 005037 001112
2828 006754
2829 006754 023727 014166 000001
2830 006762 001004
2831 006764
2832 006764 005037 014166
2833 006770
2834 006770 005237 001206
2835 006774
2836 006774
2837 006774 000137 001732

*****
*TEST 20      NOT A TEST - SEND BACK TO LOOP
*****
TST20: SCOPE
MOV      #1,$TIMES      ;;DO 1 ITERATION
                IF #BIT12 SETIN @SWR THEN
                IF PHASE2 EQ #TRUE THEN
                CMP      PHASE2,#TRUE
                BNE      $162
                TYPTXT   <<CRLF>/      ** PHASE 2 SUMMARY **/<CRLF>>
                ELSE
                BR       $163
$162:
                IF P1CNT EQ #0 THEN
                TST      P1CNT
                BNE      $164
                TYPTXT   <<CRLF>/      ** PHASE 1 SUMMARY **/<CRLF>>
                INC      P1CNT
                LET P1CNT := P1CNT + #1
                ENDIF
                ENDIF
$164:
$163:
                TYPTXT   <*<CSR: *>
                TYPOCT   DLADD
                TYPTXT   <*,VECTOR: *>
                TYPOCT   DLVEC
                TYPTXT   <*,ERRORS: *>
                TYPDEC   $ERTTL
                TYPE     , $CRLF
                ENDIF
$161:
                CLR      $ERTTL      ; RESET FOR NEXT DEVICE/PASS
                IF PHASE2 EQ #TRUE THEN
                LET PHASE2 := #0
                LET $UNIT :  $UNIT + #1
                ENDIF
$165:
                JMP      LOOP      ; BACK UP TO THE BEGINNING

```

```

2838
2839 007000 MODTST:
2840 *****
2841 *TEST 21 TEST THAT CHANNELS INTR AT ASSIGNED PRIORITY
2842 * INTERRUPTS WILL BE ENABLED ON ALL ACTIVE CHANNELS.
2843 * RECEIVER AND TRANSMITTER. THEN WE'LL CHECK TO
2844 * SEE IF THEY INTERRUPTED IN THE ASSIGNED SEQUENCE.
2845 *****
2846 007000 000004 TST21: SCOPE
2847 007002 012737 000001 001160 MOV #1,$TIMES ;;DO 1 ITERATION
2848 007010 012737 000021 001200 MOV #21,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
2849
2850
2851 ;;CLEAR OUT INTERRUPT TABLE
2852 007016 LET R0 := #INTRTABLE
2853 007016 012700 007744 MOV #INTRTABLE,R0
2854 007022 REPEAT
2855 007022 $166: LET (R0)+ := #0
2856 007022 CLR (R0)+
2857 007022 005020 UNTIL R0 EQ #TABEND
2858 007024
2859 007024 020027 007764 CMP R0,#TABEND
2860 007030 001374 BNE $166
2861
2862 ;;SET PRIORITY TO 7
2863 007032 SETPRI #PR7
2864
2865 ;;SET UP ALL INTERRUPT VECTORS
2866 007044 LET R0 := DLVEC
2867 007044 013700 001256 MOV DLVEC,R0
2868 007050 LET R1 := #RCVROSRV
2869 007050 012701 007424 MOV #RCVROSRV,R1
2870 007054 REPEAT
2871 007054 $167: LET (R0)+ : R1
2872 007054 MOV R1,(R0)+
2873 007054 010120 LET (R0)+ : #PR7
2874 007056 MOV #PR7,(R0)+
2875 007056 012720 000340 LET R1 := R1 + #30
2876 007062 ADD #30,R1
2877 007062 062701 000030 UNTIL R1 EQ #SRVEND
2878 007066
2879 007066 020127 007724 CMP R1,#SRVEND
2880 007072 001370 BNE $167
2881
2882 ;;ENABLE INTERRUPTS ON ALL ACTIVE LINES,
2883 ;;ALSO, XMIT CHAR'S TO PRIME RECEIVERS
2884
2885 ;;COPY MASK
2886 007074 LET CHMASK := MASK
2887 007074 013737 014162 007764 MOV MASK,CHMASK
2888 007102 LET CHCNT := #0
2889 007102 005037 007766 CLR CHCNT
2890 007106 LET R0 := DLADD
2891 007106 013700 001254 MOV DLADD,R0
2892 007112 REPEAT
2893 007112 $170:

```

|      |        |        |        |        |        |                 |                                        |
|------|--------|--------|--------|--------|--------|-----------------|----------------------------------------|
| 2894 | 007112 |        |        |        |        |                 | IF #BIT0 SET IN CHMASK THEN            |
| 2895 | 007112 | 032737 | 000001 | 007764 | BIT    | #BIT0,CHMASK    |                                        |
| 2896 | 007120 | 001430 |        |        | BEQ    | \$171           |                                        |
| 2897 |        |        |        |        |        |                 | ;SET RCSR IE                           |
| 2898 | 007122 |        |        |        |        |                 | LET (R0) := (R0) SET.BY #BIT6          |
| 2899 | 007122 | 052710 | 000100 |        | BIS    | #BIT6,(R0)      |                                        |
| 2900 | 007126 |        |        |        |        |                 | LET R0 := R0 + #4                      |
| 2901 | 007126 | 062700 | 000004 |        | ADD    | #4,R0           |                                        |
| 2902 |        |        |        |        |        |                 | ;SET XCSR IE                           |
| 2903 | 007132 |        |        |        |        |                 | LET (R0) := (R0) SET.BY #BIT6          |
| 2904 | 007132 | 052710 | 000100 |        | BIS    | #BIT6,(R0)      |                                        |
| 2905 |        |        |        |        |        |                 | ;LOAD XBUF                             |
| 2906 | 007136 |        |        |        |        |                 | LET 2(R0) : #252                       |
| 2907 | 007136 | 012760 | 000252 | 000002 | MOV    | #252,2(R0)      |                                        |
| 2908 |        |        |        |        |        |                 | ;GO BACK TO RCSR                       |
| 2909 | 007144 |        |        |        |        |                 | LET R0 := R0 - #4                      |
| 2910 | 007144 | 162700 | 000004 |        | SUB    | #4,R0           |                                        |
| 2911 |        |        |        |        |        |                 | ;LOOK FOR RCVR DONE                    |
| 2912 | 007150 |        |        |        |        |                 | CALL TIMER IN <#500,#RCVRDONE,R0,#SET> |
| 2913 | 007150 | 010546 |        |        | MOV    | R5,-(SP)        |                                        |
| 2914 | 007152 | 012745 | 177777 |        | MOV    | #SET,-(R5)      |                                        |
| 2915 | 007156 | 010045 |        |        | MOV    | R0,-(R5)        |                                        |
| 2916 | 007160 | 012745 | 000200 |        | MOV    | #RCVRDONE,-(R5) |                                        |
| 2917 | 007164 | 012745 | 000500 |        | MOV    | #500,-(R5)      |                                        |
| 2918 | 007170 | 004737 | 012446 |        | JSR    | PC,TIMER        |                                        |
| 2919 | 007174 | 012605 |        |        | MOV    | (SP)+,R5        |                                        |
| 2920 |        |        |        |        |        |                 | ;DID IT SET IN TIME?                   |
| 2921 | 007176 |        |        |        |        |                 | IF.ERROR THEN                          |
| 2922 | 007176 | 103001 |        |        | BCC    | \$172           |                                        |
| 2923 |        |        |        |        |        |                 | ;RCVRDONE DID NOT SET IN RCSR          |
| 2924 | 007200 |        |        |        |        |                 | ERRHRD 23                              |
| 2925 | 007200 | 104023 |        |        | ERROR  | 23              |                                        |
| 2926 | 007202 |        |        |        |        |                 | ENDIF                                  |
| 2927 | 007202 |        |        |        | \$172: |                 | ENDIF                                  |
| 2928 | 007202 |        |        |        |        |                 |                                        |
| 2929 | 007202 |        |        |        | \$171: |                 |                                        |
| 2930 |        |        |        |        |        |                 |                                        |
| 2931 | 007202 |        |        |        |        |                 | LET R0 : R0 + #10                      |
| 2932 | 007202 | 062700 | 000010 |        | ADD    | #10,R0          |                                        |
| 2933 |        |        |        |        |        |                 | ;SETUP FOR NEXT CHANNEL                |
| 2934 | 007206 |        |        |        |        |                 | LET CHMASK := CHMASK SHIFT -1          |
| 2935 | 007206 | 006237 | 007764 |        | ASR    | CHMASK          |                                        |
| 2936 | 007212 |        |        |        |        |                 | LET CHCNT : CHCNT + #1                 |
| 2937 | 007212 | 005237 | 007766 |        | INC    | CHCNT           |                                        |
| 2938 | 007216 |        |        |        |        |                 | IF CHCNT EQ #3 THEN                    |
| 2939 | 007216 | 023727 | 007766 | 000003 | CMP    | CHCNT,#3        |                                        |
| 2940 | 007224 | 001007 |        |        | BNE    | \$173           |                                        |
| 2941 | 007226 |        |        |        |        |                 | CALL TSTCON                            |
| 2942 | 007226 | 004737 | 013530 |        | JSR    | PC,TSTCON       |                                        |
| 2943 | 007232 |        |        |        |        |                 | IF CONSOLE NE #0 THEN                  |
| 2944 | 007232 | 005737 | 013604 |        | TST    | CONSOLE         |                                        |
| 2945 | 007236 | 001402 |        |        | BEQ    | \$174           |                                        |
| 2946 | 007240 |        |        |        |        |                 | LET CHCNT := CHCNT + #1                |
| 2947 | 007240 | 005237 | 007766 |        | INC    | CHCNT           |                                        |
| 2948 | 007244 |        |        |        |        |                 | ENDIF                                  |
| 2949 | 007244 |        |        |        | \$174: |                 |                                        |

```

2950 007244                                ENDIF
2951 007244                                $173:
2952 007244                                UNTIL CHCNT EQ #4
2953 007244 023727 007766 000004          CMP    CHCNT,#4
2954 007252 001317                      BNE     $170
2955
2956                                ;ALL XMIT & REC INTERRUPTS SHOULD BE
2957                                ;IN CONTENTION.
2958                                ;RECEIVE INTERRUPTS HAVE PRIORITY
2959                                ;OVER XMIT INTERRUPTS.
2960                                ;LOW CHANNELS HAVE PRIORITY OVER HIGH CHANNELS.
2961                                ;THEREFORE, ONCE CPU PRIORITY IS LOWERED,
2962                                ;INTERRUPTS S/B IN THE ORDER SHOWN IN THE
2963                                ;CHANNEL IDENTIFIER TABLE (RCHO:)
2964
2965                                ;SET UP POINTER FOR SERVICE ROUTINE
2966 007254                                LET R4 := #INTRTABLE
2967 007254 012704 007744          MOV     #INTRTABLE,R4
2968
2969                                ;LET EM GO
2970 007260          SETPRI #PRO
2971
2972                                ;DISABLE ALL INTERRUPTS
2973 007272          BRESET
2974 007272 000005          RESET
2975
2976 007274                                LET TCNT := #0
2977 007274 005037 007772          CLR     TCNT
2978
2979                                ;CLEAR ERR FLAG
2980 007300                                LET PRIERR := #0
2981 007300 005037 007770          CLR     PRIERR
2982 007304 012700 007744          MOV     #INTRTABLE,R0
2983
2984                                ;SETUP EXPECTED VALUE
2985 007310                                LET R1 := #INTRTABLE+2
2986 007314 013737 014162 007764          MOV     #INTRTABLE+2,R1
2987                                MOV     MASK,CHMASK
2988                                ;GET # OF CHANNELS IN TCNT
2989 007322 006237 007764          1$:    ASR     CHMASK
2990 007326 103002                      BCC     2$
2991 007330 005237 007772          INC     TCNT
2992 007334 005737 007764          2$:    TST     CHMASK
2993 007340 001370                      BNE     1$
2994
2995                                ;CORRECT CHANNEL CT FOR CONSOLE
2996 007342 004737 013530          JSR     PC,TSTCON
2997 007346 005737 013604          TST     CONSOLE
2998 007352 001402                      BEQ     3$
2999 007354 005337 007772          DEC     TCNT
3000 007360 006337 007772          3$:    ASL     TCNT
3001                                ;APPLY X2 MULT FACTOR
3002 007364 005337 007772          4$:    DEC     TCNT
3003 007370 001410                      BEQ     5$
3004 007372 023727 007772 177777          CMP     TCNT,#-1
3005 007400 001404                      BEQ     5$
3006 007402 022120                      CMP     (R1)+,(R0)+
3007 007404 003367                      BGT     4$
3008                                ;ALL 1'S ?
3009                                ;BR IF YES
3010                                ;R1-R0
3011                                ;BR IF R1>R0

```

CVDLAAO DLV11-J TEST MACY11 30G(1063) 17-NOV-78 09:39 PAGE 72 6  
CVDLAB.P11 17-NOV-78 09:38 T21 TEST THAT CHANNELS INTR AT ASSIGNED PRIORITY

SEQ 0071

```
3006 007406 005237 007770      INC      PRIERR      ;ELSE ERROR
3007
3008 007412      5$:
3009 007412
3010 007412 005737 007770      TST      PRIERR
3011 007416 001401      BEQ      $175
3012 007420
3013 007420 104250      ERROR    250
3014
3015
3016
3017
3018
3019 007422
3020 007422      $175:
3021
3022 007422
3023 007422 000565      BR      TST22

;CHANNELS DID NOT INTR ACCORDING TO
;ASSIGNED PRIORITY. THE BYTE ENTRIES
;IN INTRTABLE: SHOULD BE IN THE ORDER
;THAT THEY APPEAR IN THE CHANNEL #
;TABLE .(EXCLUDING CHANNELS NOT ACTIVE)
ENDIF
EXIT
;;EXIT THIS TEST
```

```

3024
3025
3026
3027
3028
3029 007424
3030 007424
3031
3032 007424
3033 007424 013724 007724
3034
3035 007430
3036 007430 013737 001260 007774
3037 007436
3038 007436 062737 000000 007774
3039
3040 007444
3041 007444 042777 000100 000322
3042 007452
3043 007452 000002
3044
3045
3046 007454
3047 007454
3048
3049 007454
3050 007454 013724 007734
3051
3052 007460
3053 007460 013737 001260 007774
3054 007466
3055 007466 062737 000004 007774
3056
3057 007474
3058 007474 042777 000100 000272
3059 007502
3060 007502 000002
3061
3062 007504
3063 007504
3064
3065 007504
3066 007504 013724 007726
3067
3068 007510
3069 007510 013737 001260 007774
3070 007516
3071 007516 062737 000010 007774
3072
3073 007524
3074 007524 042777 000100 000242
3075 007532
3076 007532 000002
3077
3078 007534
3079 007534

*****
*                               *
*          START OF SERVICE ROUTINES          *
*                               *
*****

RCVROSRV:
BGNSRV RCVROSRV
;PUT INTR IDENTIFIER IN INTRTABLE
LET (R4)+ := RCH0
;GENERATE CSR ADDRESS
LET TEMP := RCSR
LET TEMP := TEMP + #0
;ONE INTR IS ALL WE WANT FROM HERE
LET @TEMP := @TEMP CLR.BY #BIT6
ENDSRV

XMITOSRV:
BGNSRV XMITOSRV
;PUT INTR IDENTIFIER IN INTRTABLE
LET (R4)+ := TCH0
;GENERATE CSR ADDRESS
LET TEMP := RCSR
LET TEMP := TEMP + #4
;ONE INTR IS ALL WE WANT FROM HERE
LET @TEMP := @TEMP CLR.BY #BIT6
ENDSRV

RCVR1SRV:
BGNSRV RCVR1SRV
;PUT INTR IDENTIFIER IN INTRTABLE
LET (R4)+ := RCH1
;GENERATE CSR ADDRESS
LET TEMP := RCSR
LET TEMP := TEMP + #10
;ONE INTR IS ALL WE WANT FROM HERE
LET @TEMP := @TEMP CLR.BY #BIT6
ENDSRV

XMIT1SRV:
BGNSRV XMIT1SRV

```

```

3080                                     ;PUT INTR IDENTIFIER IN INTRTABLE
3081 007534                                     LET (R4)+ := TCH1
3082 007534 013724 007736                 MOV    TCH1,(R4)+
3083                                     ;GENERATE CSR ADDRESS
3084 007540                                     LET TEMP := RCSR
3085 007540 013737 001260 007774         MOV    RCSR,TEMP
3086 007546                                     LET TEMP := TEMP + #14
3087 007546 062737 000014 007774         ADD    #14,TEMP
3088                                     ;ONE INTR IS ALL WE WANT FROM HERE
3089 007554                                     LET @TEMP := @TEMP CLR.BY #BIT6
3090 007554 042777 000100 000212         BIC    #BIT6,@TEMP
3091 007562                                     ENDSRV
3092 007562 000002                                     RTI
3093
3094 007564                                     BGNSRV RCVR2SRV
3095 007564
3096                                     ;PUT INTR IDENTIFIER IN INTRTABLE
3097 007564                                     LET (R4)+ := RCH2
3098 007564 013724 007730                 MOV    RCH2,(R4)+
3099                                     ;GENERATE CSR ADDRESS
3100 007570                                     LET TEMP := RCSR
3101 007570 013737 001260 007774         MOV    RCSR,TEMP
3102 007576                                     LET TEMP := TEMP + #20
3103 007576 062737 000020 007774         ADD    #20,TEMP
3104                                     ;ONE INTR IS ALL WE WANT FROM HERE
3105 007604                                     LET @TEMP := @TEMP CLR.BY #BIT6
3106 007604 042777 000100 000162         BIC    #BIT6,@TEMP
3107 007612                                     ENDSRV
3108 007612 000002                                     RTI
3109
3110 007614                                     BGNSRV XMIT2SRV
3111 007614
3112                                     ;PUT INTR IDENTIFIER IN INTRTABLE
3113 007614                                     LET (R4)+ := TCH2
3114 007614 013724 007740                 MOV    TCH2,(R4)+
3115                                     ;GENERATE CSR ADDRESS
3116 007620                                     LET TEMP := RCSR
3117 007620 013737 001260 007774         MOV    RCSR,TEMP
3118 007626                                     LET TEMP := TEMP + #24
3119 007626 062737 000024 007774         ADD    #24,TEMP
3120                                     ;ONE INTR IS ALL WE WANT FROM HERE
3121 007634                                     LET @TEMP := @TEMP CLR.BY #BIT6
3122 007634 042777 000100 000132         BIC    #BIT6,@TEMP
3123 007642                                     ENDSRV
3124 007642 000002                                     RTI
3125
3126 007644                                     BGNSRV RCVR3SRV
3127 007644
3128                                     ;PUT INTR IDENTIFIER IN INTRTABLE
3129 007644                                     LET (R4)+ := RCH3
3130 007644 013724 007732                 MOV    RCH3,(R4)+
3131                                     ;GENERATE CSR ADDRESS
3132 007650                                     LET TEMP := RCSR
3133 007650 013737 001260 007774         MOV    RCSR,TEMP
3134 007656                                     LET TEMP := TEMP + #30
3135 007656 062737 000030 007774         ADD    #30,TEMP
  
```

```

3136                                     ;ONE INTR IS ALL WE WANT FROM HERE
3137 007664                               LET @TEMP := @TEMP CLR.BY #BIT6
3138 007664 042777 000100 000102        BIC    #BIT6,@TEMP
3139 007672                               ENDSRV
3140 007672 000002                       RTI
3141
3142 007674                               BGNSRV XMIT3SRV
3143 007674                               XMIT3SRV:
3144                                     ;PUT INTR IDENTIFIER IN INTRTABLE
3145 007674                               LET (R4)+ := TCH3
3146 007674 013724 007742                MOV    TCH3,(R4)+
3147                                     ;GENERATE CSR ADDRESS
3148 007700                               LET TEMP := RCSR
3149 007700 013737 001260 007774        MOV    RCSR,TEMP
3150 007706                               LET TEMP := TEMP + #34
3151 007706 062737 000034 007774        ADD    #34,TEMP
3152                                     ;ONE INTR IS ALL WE WANT FROM HERE
3153 007714                               LET @TEMP : @TEMP CLR.BY #BIT6
3154 007714 042777 000100 000052        BIC    #BIT6,@TEMP
3155 007722                               ENDSRV
3156 007722 000002                       RTI
3157 007724                               SRVEND:
  
```

3158  
3159  
3160  
3161  
3162  
3163  
3164  
3165  
3166 007724 000000  
3167 007726 000001  
3168 007730 000002  
3169 007732 000003  
3170 007734 000010  
3171 007736 000011  
3172 007740 000012  
3173 007742 000013  
3174  
3175  
3176  
3177  
3178  
3179  
3180 007744 000010  
3181 007764  
3182  
3183 007764 000000  
3184 007766 000000  
3185 007770 000000  
3186 007772 000000  
3187 007774 000000  
3188

\*\*\*CHANNEL IDENTIFIER TABLE\*\*  
:THE ENTRIES FROM THIS  
:TABLE ARE PLACED IN THE INTRTABLE  
:IN THE ORDER THAT THE INTR'S  
:OCCUR. (EXCLUDING NON-ACTIVE CH'S)

RCH0: .WORD 0  
RCH1: .WORD 1  
RCH2: .WORD 2  
RCH3: .WORD 3  
TCH0: .WORD 10  
TCH1: .WORD 11  
TCH2: .WORD 12  
TCH3: .WORD 13

:THIS TABLE WILL CONTAIN ENTRIES  
:REPRESENTING EACH INTR IN THE ORDER  
:THAT IT OCCURED  
: S/B IN THE ABOVE ORDER

INTRTABLE: .BLKW 8.  
TABEND:

CHMASK: .WORD 0 :BITS 3-0 INDICATE WHICH CH'S ARE ACTIVE FOR THIS TEST  
CHCNT: .WORD 0 :CHANNEL COUNTER  
PRIERR: .WORD 0 :ERROR WORD, CONTAINS NO. OF INTR OUT OF ORDER  
TCNT: .WORD 0 :TABLE PASS CTR  
TEMP: .WORD 0 :FOR SERVICE ROUTINES

```

3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202 007776 000004
3203 010000 012737 000001 001160
3204 010006 012737 000022 001200
3205
3206 010C14
3207 010C14 005037 012366
3208
3209
3210
3211 010020
3212 010020 012700 011360
3213 010024
3214 010024
3215 010024
3216 010024 005020
3217 010026
3218 010026 020027 012360
3219 010032 001374
3220
3221
3222 010034
3223 010034 012701 010660
3224 010040
3225 010040 013700 001256
3226 010044
3227 010044
3228 010044
3229 010044 010120
3230 010046
3231 010046 012720 000340
3232 010052
3233 010052 062701 000054
3234 010056
3235 010056 010120
3236 010060
3237 010060 012720 000340
3238 010064
3239 010064 062701 000044
3240 010070
3241 010070 020127 011360
3242 010074 001363
3243
3244 010C76
  
```

\*\*\*\*\*  
 TEST 22 TEST DATA TRANSFERS WITH ALL ACTIVE LINES INTERRUPTING  
 IN THIS WE'LL ENABLE INTERRUPTS ON ALL CHANNELS  
 FOR THIS DLV11-J. THEN WE'LL XMIT AN INCREMENTING  
 DATA PATERN VIA INTERUPTS AND RECORD THE RECEIVER  
 INTR. IN THE RECEIVER STATUS TABLE.  
 NOTE: DOUBLE BUFFERING CANNOT BE TESTED AT ITS MAX SPEED  
 BECAUSE OF APT CONSIDERATIONS. I.E. APT SENDS  
 'BREAKS' WHICH CAUSE OVERRUN ERRORS. THEREFORE  
 THE XMIT IE IS NOT ENABLED AGAIN UNTIL THE RECVR  
 HAS OBTAINED THE PREVIOUS WORD.  
 \*\*\*\*\*

```

TST22: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
MOV #22,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
;;CLR ERROR WORD,
LET ERWRD : #0

;;CLEAR OUT RECEIVER TABLES
LET R0 := #CHOTAB
REPEAT
LET (R0)+ : #0
UNTIL R0 EQ #STATEND

;;SET UP ALL VECTORS
LET R1 := #ROSRV
LET R0 := DLVEC
REPEAT
LET (R0)+ := R1
LET (R0)+ := #PR7
LET R1 := R1 + #54
LET (R0)+ := R1
LET (R0)+ := #PR7
LET R1 := R1 + #44
UNTIL R1 EQ #SEREND

;;SET PRIORITY TO 7
SETPRI #PR7
  
```

\$176:

```

CLR (R0)+
CMP R0,#STATEND
BNE $176
  
```

\$177:

```

MOV R1,(R0)+
MOV #PR7,(R0)+
ADD #54,R1
MOV R1,(R0)+
MOV #PR7,(R0)+
ADD #44,R1
CMP R1,#SEREND
BNE $177
  
```

```

3245
3246                                     :ENABLE INTR'S ON ALL ACTIVE LINES
3247                                     :THIS MODULE
3248 010110                                LET RO := DLADD
3249 010110 013700 001254                MOV    DLADD,RO
3250                                     :INITIALIZE
3251 010114                                LET CHMSK := MASK
3252 010114 013737 014162 012360        MOV    MASK,CHMSK
3253 010122                                LET CHCTR := #0
3254 010122 005037 012362                CLR    CHCTR
3255                                     :ACTIVE CHANNEL CTR
3256 010126                                LET ACTCH := #0
3257 010126 005037 012364                CLR    ACTCH
3258 010132                                REPEAT
3259 010132                                IF #BIT0 SET IN CHMSK THEN
3260                                     :# OF ACTIVE CH
3261 010132 032737 000001 012360        BIT     #BIT0,CHMSK
3262 010140 001413                        BEQ     $201
3263                                     :# OF ACTIVE CH
3264 010142                                LET ACTCH := ACTCH + #1
3265 010142 005237 012364                INC     ACTCH
3266                                     :SET RCSR IE
3267 010146                                LET (RO) := (RO) SET.BY #BIT6
3268 010146 052710 000100                BIS     #BIT6,(RO)
3269 010152                                LET RO := RO + #4
3270 010152 062700 000004                ADD     #4,RO
3271                                     :SET XCSR IE
3272 010156                                LET (RO) := (RO) SET.BY #BIT6
3273 010156 052710 000100                BIS     #BIT6,(RO)
3274 010162                                LET RO := RO + #4
3275 010162 062700 000004                ADD     #4,RO
3276 010166                                ELSE
3277 010166 000402                        BR      $202
3278 010170                                LEI    RO := RO + #10
3279 010170                                ADD     #10,RO
3280 010170 062700 000010                ADD     #10,RO
3281 010174                                ENDIF
3282 010174                                LET CHMSK := CHMSK SHIFT -1
3283 010174                                $201:
3284 010174 006237 012360                ASR     CHMSK
3285
3286                                     LET CHCTR := CHCTR + #1
3287 010200                                IF CHCTR EQ #3 THEN
3288 010200 005237 012362                INC     CHCTR
3289 010204                                CALL TSTCON
3290 010204 023727 012362 000003        CMP     CHCTR,#3
3291 010212 001007                        BNE     $203
3292 010214                                CALL TSTCON
3293 010214 004737 013530                JSR     PC,TSTCON
3294 010220                                IF CONSOLE NE #0 THEN
3295 010220 005737 013604                TST     CONSOLE
3296 010224 001402                        BEQ     $204
3297 010226                                LET CHCTR := CHCTR + #1
3298 010226 005237 012362                INC     CHCTR
3299 010232                                ENDIF
3300 010232                                $204:

```

```

3301 010232
3302 010232
3303 010232
3304 010232 023727 012362 000004
3305 010240 001334
3306
3307
3308 010242
3309 010242 012700 011360
3310 010246
3311 010246 012701 011560
3312 010252
3313 010252 012702 011760
3314 010256
3315 010256 012703 012160
3316
3317
3318
3319
3320 010262
3321 010262 005037 012372
3322
3323 010266
3324 010266 005004
3325
3326 010270
3327 010270 012705 012374
3328 010274
3329 010274
3330 010274
3331 010274 032737 000001 001220
3332 010302 001005
3333 010304
3334 010304 016425 012426
3335 010310
3336 010310 016425 012426
3337 010314
3338 010314 000404
3339 010316
3340 010316
3341 010316 016425 012436
3342 010322
3343 010322 016425 012436
3344 010326
3345 010326
3346 010326
3347 010326 062704 000002
3348 010332
3349 010332 020427 000010
3350 010336 001356
3351
3352
3353 010340
3354 010340 013737 012364 012414
3355
3356 010346

$203:
      CMP      CHCTR,#4
      BNE      $200

      ;INIT BUFFER POINTERS FOR SERVICE ROUTINES
      LET R0 := #CH0TAB
      LET R1 := #CH1TAB
      LET R2 := #CH2TAB
      LET R3 := #CH3TAB

      ;FILL XBUF WORDS
      ;FROM TMP0 THRU TMP3E
      ;START WORD CH 0 ALWAYS 0
      LET TMP0 := #0

      ;OFFSET TO INDEX
      LET R4 := #0

      ;PTR TO TABLE TO BE FILLED
      LET R5 := #TMPOE
      REPEAT
        IF #BIT0 NOTSET IN $USWR THEN
          LET (R5)+ := TABL7(R4)
          LET (R5)+ := TABL7(R4)
        ELSE
          LET (R5)+ := TABL8(R4)
          LET (R5)+ := TABL8(R4)
        ENDIF
      UNTIL R4 EQ #10

      LET R4 := R4 + #2

      ;INIT TMP4
      ;WILL BE DECR BY XMIT SERVICE ROUTINES TILL 0
      LET TMP4 := ACTCH

      ;SET PRIORITY TO 0
      SETPRI #PRO

$205:
      BIT      #BIT0,$USWR
      BNE      $206

      MOV      TABL7(R4),(R5)+
      MOV      TABL7(R4),(R5)+
      BR       $207

$206:
      MOV      TABL8(R4),(R5)+
      MOV      TABL8(R4),(R5)+
      ENDIF

$207:
      ADD      #2,R4
      UNTIL R4 EQ #10

      CMP      R4,#10
      BNE      $205
  
```

| Address | Op Code | Op 1 | Op 2 | Op 3 | Op 4 | Op 5 | Op 6 | Op 7 | Op 8 | Op 9 | Op 10 | Op 11 | Op 12 | Op 13 | Op 14 | Op 15 | Op 16 | Op 17 | Op 18 | Op 19 | Op 20 | Op 21 | Op 22 | Op 23 | Op 24 | Op 25 | Op 26 | Op 27 | Op 28 | Op 29 | Op 30 | Op 31 | Op 32 | Op 33 | Op 34 | Op 35 | Op 36 | Op 37 | Op 38 | Op 39 | Op 40 | Op 41 | Op 42 | Op 43 | Op 44 | Op 45 | Op 46 | Op 47 | Op 48 | Op 49 | Op 50 | Op 51 | Op 52 | Op 53 | Op 54 | Op 55 | Op 56 | Op 57 | Op 58 | Op 59 | Op 60 | Op 61 | Op 62 | Op 63 | Op 64 | Op 65 | Op 66 | Op 67 | Op 68 | Op 69 | Op 70 | Op 71 | Op 72 | Op 73 | Op 74 | Op 75 | Op 76 | Op 77 | Op 78 | Op 79 | Op 80 | Op 81 | Op 82 | Op 83 | Op 84 | Op 85 | Op 86 | Op 87 | Op 88 | Op 89 | Op 90 | Op 91 | Op 92 | Op 93 | Op 94 | Op 95 | Op 96 | Op 97 | Op 98 | Op 99 | Op 100 | Op 101 | Op 102 | Op 103 | Op 104 | Op 105 | Op 106 | Op 107 | Op 108 | Op 109 | Op 110 | Op 111 | Op 112 | Op 113 | Op 114 | Op 115 | Op 116 | Op 117 | Op 118 | Op 119 | Op 120 | Op 121 | Op 122 | Op 123 | Op 124 | Op 125 | Op 126 | Op 127 | Op 128 | Op 129 | Op 130 | Op 131 | Op 132 | Op 133 | Op 134 | Op 135 | Op 136 | Op 137 | Op 138 | Op 139 | Op 140 | Op 141 | Op 142 | Op 143 | Op 144 | Op 145 | Op 146 | Op 147 | Op 148 | Op 149 | Op 150 | Op 151 | Op 152 | Op 153 | Op 154 | Op 155 | Op 156 | Op 157 | Op 158 | Op 159 | Op 160 | Op 161 | Op 162 | Op 163 | Op 164 | Op 165 | Op 166 | Op 167 | Op 168 | Op 169 | Op 170 | Op 171 | Op 172 | Op 173 | Op 174 | Op 175 | Op 176 | Op 177 | Op 178 | Op 179 | Op 180 | Op 181 | Op 182 | Op 183 | Op 184 | Op 185 | Op 186 | Op 187 | Op 188 | Op 189 | Op 190 | Op 191 | Op 192 | Op 193 | Op 194 | Op 195 | Op 196 | Op 197 | Op 198 | Op 199 | Op 200 | Op 201 | Op 202 | Op 203 | Op 204 | Op 205 | Op 206 | Op 207 | Op 208 | Op 209 | Op 210 | Op 211 | Op 212 | Op 213 | Op 214 | Op 215 | Op 216 | Op 217 | Op 218 | Op 219 | Op 220 | Op 221 | Op 222 | Op 223 | Op 224 | Op 225 | Op 226 | Op 227 | Op 228 | Op 229 | Op 230 | Op 231 | Op 232 | Op 233 | Op 234 | Op 235 | Op 236 | Op 237 | Op 238 | Op 239 | Op 240 | Op 241 | Op 242 | Op 243 | Op 244 | Op 245 | Op 246 | Op 247 | Op 248 | Op 249 | Op 250 | Op 251 | Op 252 | Op 253 | Op 254 | Op 255 | Op 256 | Op 257 | Op 258 | Op 259 | Op 260 | Op 261 | Op 262 | Op 263 | Op 264 | Op 265 | Op 266 | Op 267 | Op 268 | Op 269 | Op 270 | Op 271 | Op 272 | Op 273 | Op 274 | Op 275 | Op 276 | Op 277 | Op 278 | Op 279 | Op 280 | Op 281 | Op 282 | Op 283 | Op 284 | Op 285 | Op 286 | Op 287 | Op 288 | Op 289 | Op 290 | Op 291 | Op 292 | Op 293 | Op 294 | Op 295 | Op 296 | Op 297 | Op 298 | Op 299 | Op 300 | Op 301 | Op 302 | Op 303 | Op 304 | Op 305 | Op 306 | Op 307 | Op 308 | Op 309 | Op 310 | Op 311 | Op 312 | Op 313 | Op 314 | Op 315 | Op 316 | Op 317 | Op 318 | Op 319 | Op 320 | Op 321 | Op 322 | Op 323 | Op 324 | Op 325 | Op 326 | Op 327 | Op 328 | Op 329 | Op 330 | Op 331 | Op 332 | Op 333 | Op 334 | Op 335 | Op 336 | Op 337 | Op 338 | Op 339 | Op 340 | Op 341 | Op 342 | Op 343 | Op 344 | Op 345 | Op 346 | Op 347 | Op 348 | Op 349 | Op 350 | Op 351 | Op 352 | Op 353 | Op 354 | Op 355 | Op 356 | Op 357 | Op 358 | Op 359 | Op 360 | Op 361 | Op 362 | Op 363 | Op 364 | Op 365 | Op 366 | Op 367 | Op 368 | Op 369 | Op 370 | Op 371 | Op 372 | Op 373 | Op 374 | Op 375 | Op 376 | Op 377 | Op 378 | Op 379 | Op 380</ |
|---------|---------|------|------|------|------|------|------|------|------|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|----------|
|---------|---------|------|------|------|------|------|------|------|------|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|----------|

C 7

CVDLAAO DLV11-J TEST MACY11 30G(1063) 17-NOV-78 09:39 PAGE 81  
 CVDLAB.P11 17-NOV-78 09:38 T22 TEST DATA TRANSFERS WITH ALL ACTIVE LINES INTERRUPTING SEQ 0080

|      |        |        |               |        |               |                                 |
|------|--------|--------|---------------|--------|---------------|---------------------------------|
| 3413 | 010500 |        |               |        |               | LET R0 := R0 + #2               |
| 3414 | 010500 | 062700 | 000002        | ADD    | #2,R0         |                                 |
| 3415 |        |        |               |        |               | ;GET MAX WORD FOR THAT CH.      |
| 3416 | 010504 |        |               |        |               | LET R3 := TMP0(R0)              |
| 3417 | 010504 | 016003 | 012372        | MOV    | TMP0(R0),R3   |                                 |
| 3418 |        |        |               |        |               |                                 |
| 3419 | 010510 |        |               |        |               | REPEAT                          |
| 3420 | 010510 |        |               | \$214: |               |                                 |
| 3421 | 010510 |        |               |        |               | IF #BIT15 SET IN (R2) THEN      |
| 3422 | 010510 | 032712 | 100000        | BIT    | #BIT15,(R2)   |                                 |
| 3423 | 010514 | 001403 |               | BEQ    | \$215         |                                 |
| 3424 | 010516 |        |               |        |               | LET ERWRD := ERWRD SET BY #BIT8 |
| 3425 | 010516 | 052737 | 000400 012366 | BIS    | #BIT8,ERWRD   |                                 |
| 3426 | 010524 |        |               |        |               | ENDIF                           |
| 3427 | 010524 |        |               | \$215: |               |                                 |
| 3428 |        |        |               |        |               | ;CHECK FOR DATA COMPARE ERROR   |
| 3429 | 010524 |        |               |        |               | IFB (R2) NE R1 THEN             |
| 3430 | 010524 | 121201 |               | CMPS   | (R2),R1       |                                 |
| 3431 | 010526 | 001402 |               | BEQ    | \$216         |                                 |
| 3432 | 010530 |        |               |        |               | LET ERWRD := ERWRD + #1         |
| 3433 | 010530 | 005237 | 012366        | INC    | ERWRD         |                                 |
| 3434 | 010534 |        |               |        |               | ENDIF                           |
| 3435 | 010534 |        |               | \$216: |               |                                 |
| 3436 |        |        |               |        |               | ;BUMP EXPECTED DATA WORD        |
| 3437 | 010534 |        |               |        |               | LET R1 := R1 + #1               |
| 3438 | 010534 | 005201 |               | INC    | R1            |                                 |
| 3439 |        |        |               |        |               | ;BUMP TABLE PTR                 |
| 3440 | 010536 |        |               |        |               | LET R2 := R2 + #2               |
| 3441 | 010536 | 062702 | 000002        | ADD    | #2,R2         |                                 |
| 3442 |        |        |               |        |               |                                 |
| 3443 | 010542 |        |               |        |               | UNTIL R1 EQ R3 OR ERWRD NE #0   |
| 3444 | 010542 | 020103 |               | CMP    | R1,R3         |                                 |
| 3445 | 010544 | 001403 |               | BEQ    | \$217         |                                 |
| 3446 | 010546 | 005737 | 012366        | TST    | ERWRD         |                                 |
| 3447 | 010552 | 001756 |               | BEQ    | \$214         |                                 |
| 3448 | 010554 |        |               | \$217: |               |                                 |
| 3449 | 010554 |        |               |        |               | ENDIF                           |
| 3450 | 010554 |        |               | \$213: |               |                                 |
| 3451 | 010554 |        |               |        |               | LET CHCTR := CHCTR + #1         |
| 3452 | 010554 | 005237 | 012362        | INC    | CHCTR         |                                 |
| 3453 | 010560 |        |               |        |               | LET CHMSK : CHMSK SHIFT -1      |
| 3454 | 010560 | 006237 | 012360        | ASR    | CHMSK         |                                 |
| 3455 | 010564 |        |               |        |               | IF CHCTR EQ #3 THEN             |
| 3456 | 010564 | 023727 | 012362 000003 | CMP    | CHCTR,#3      |                                 |
| 3457 | 010572 | 001010 |               | BNE    | \$220         |                                 |
| 3458 | 010574 |        |               |        |               | CALL TSTCON                     |
| 3459 | 010574 | 004737 | 013530        | JSR    | PC,TSTCON     |                                 |
| 3460 | 010600 |        |               |        |               | IF CONSOLE EQ #TRUE THEN        |
| 3461 | 010600 | 023727 | 013604 000001 | CMP    | CONSOLE,#TRUE |                                 |
| 3462 | 010606 | 001002 |               | BNE    | \$221         |                                 |
| 3463 |        |        |               |        |               | ;EXIT                           |
| 3464 | 010610 |        |               |        |               | LET CHCTR := CHCTR + #1         |
| 3465 | 010610 | 005237 | 012362        | INC    | CHCTR         |                                 |
| 3466 | 010614 |        |               |        |               | ENDIF                           |
| 3467 | 010614 |        |               | \$221: |               |                                 |
| 3468 | 010614 |        |               |        |               | ENDIF                           |

```

3469 010614 $220:
3470 010614 UNTIL CHCTR EQ #4 OR ERWRD NE #0
3471 010614 023727 012362 000004 CMP CHCTR,#4
3472 010622 001403 BEQ $222
3473 010624 005737 012366 TST ERWRD
3474 010630 001707 BEQ $212
3475 010632 $222:
3476 010632 IF #BIT8 SET IN ERWRD THEN
3477 010632 032737 000400 012366 BIT #BIT8,ERWRD
3478 010632 001401 BEQ $223
3479 010640
3480 ;ERROR FLAG UP AFTER TRANSFER
3481 010642 ERRHRD 270
3482 010642 104270 ERROR 270
3483 010644 ENDF
3484 010644 $223:
3485 010644 IFB ERWRD NE #0 THEN
3486 010644 105737 012366 TSTB ERWRD
3487 010650 001401 BEQ $224
3488 ;DATA COMPARE ERROR
3489 010652 ERRHRD 271
3490 010652 104271 ERROR 271
3491 010654 ENDF
3492 010654 $224:
3493
3494
3495 010654 000137 006502 JMP TST20 ;EXIT TEST
  
```

```

3496
3497
3498
3499
3500
3501 010660
3502 010660
3503 010660
3504 010660 013737 001260 012370
3505 010666
3506 010666 062737 000002 012370
3507 010674
3508 010674 017720 001470
3509 010700
3510 010700 023737 012372 012374
3511 010706 001407
3512
3513 010710
3514 010710 062737 000002 012370
3515
3516 010716
3517 010716 052777 000100 001444
3518 010724
3519 010724 000402
3520 010726
3521
3522 010726
3523 010726 005337 012414
3524 010732
3525 010732
3526 010732
3527 010732 000002
3528
3529 010734
3530 010734
3531 010734
3532 010734 013737 001260 012370
3533 010742
3534 010742 062737 000006 012370
3535 010750
3536 010750 013777 012372 001412
3537 010756
3538 010756 005237 012372
3539
3540 010762
3541 010762 162737 000002 012370
3542
3543
3544 010770
3545 010770 042777 000100 001372
3546 010776
3547 010776 000002
3548
3549 011000
3550 011000
3551 011000

*****
* START OF SERVICE ROUTINES
*****

R0SRV:
      MOV      RCSR,TMP
      ADD      #2,TMP
      MOV      @TMP,(R0)+
      CMP      TMP0,TMPOE
      BEQ      $225
      ;GO TO XCSR
      LET TMP : TMP + #2
      ;ENABLE XMIT INTERR
      LET @TMP : @TMP SET.BY #BIT6
      ELSE
      ;ALL DONE
      LET TMP4 :- TMP4 - #1
      ENDIF
      ENDSRV

$225:
      DEC      TMP4
      $226:
      RTI

X0SRV:
      MOV      RCSR,TMP
      ADD      #6,TMP
      MOV      TMP0,@TMP
      INC      TMP0
      ;GO BACK TO XCSR
      LET TMP := TMP - #2
      ;DISABLE XMIT INTERRUPTS,
      ;NOT EXER DOUBLE BUFFERING
      LET @TMP := @TMP CLR.BY #BIT6
      ENDSRV

R1SRV:
      LET TMP := RCSR
  
```

```

3552 011000 013737 001260 012370      MOV      RCSP,TMP
3553 011006                                LET TMP := TMP + #12
3554 011006 062737 000012 012370      ADD      #12,TMP
3555 011014                                LET (R1)+ := @TMP
3556 011014 017721 001350      MOV      @TMP,(R1)+
3557 011020                                IF TMP1 NE TMP1E THEN
3558 011020 023737 012376 012400      CMP      TMP1,TMP1E
3559 011026 001407      BEQ      $227
3560                                ;GO TO XCSR
3561 011030                                LET TMP : TMP + #2
3562 011030 062737 000002 012370      ADD      #2,TMP
3563                                ;ENABLE XMIT INTERRU
3564 011036                                LET @TMP := @TMP SET.BY #BIT6
3565 011036 052777 000100 001324      BIS      #BIT6,@TMP
3566 011044                                ELSE
3567 011044 000402      BR      $230
3568 011046                                ;ALL DONE
3569                                LET TMP4 : TMP4 - #1
3570 011046                                ENDIF
3571 011046 005337 012414      DEC      TMP4
3572 011052                                ENDSRV
3573 011052                                $230:
3574 011052 000002      RTI
3575 011052
3576
3577 011054                                BGNSRV X1SRV
3578 011054                                LET TMP : RCSR
3579 011054                                LET TMP : TMP + #16
3580 011054 013737 001260 012370      MOV      RCSR,TMP
3581 011062                                LET @TMP : TMP1
3582 011062 062737 000016 012370      ADD      #16,TMP
3583 011070                                LET @TMP : TMP1
3584 011070 013777 012376 001272      MOV      TMP1,@TMP
3585 011076                                LET TMP1 :- TMP1 + #1
3586 011076 005237 012376      INC      TMP1
3587                                ;GO BACK TO XCSR
3588 011102                                LET TMP := TMP - #2
3589 011102 162737 000002 012370      SUB      #2,TMP
3590                                ;DISABLE XMIT INTERRUPTS,
3591                                ;NOT EXER DOUBLE BUFF
3592 011110                                LET @TMP := @TMP CLR.BY #BIT6
3593 011110 042777 000100 001252      BIC      #BIT6,@TMP
3594 011116                                ENDSRV
3595 011116 000002      RTI
3596
3597 011120                                BGNSRV R2SRV
3598 011120                                LET TMP :- RCSR
3599 011120                                LET TMP :- TMP + #22
3600 011120 013737 001260 012370      MOV      RCSR,TMP
3601 011126                                LET TMP :- TMP + #22
3602 011126 062737 000022 012370      ADD      #22,TMP
3603 011134                                LET (R2)+ :- @TMP
3604 011134 017722 001230      MOV      @TMP,(R2)+
3605 011140                                IF TMP2 NE TMP2E THEN
3606 011140 023737 012402 012404      CMP      TMP2,TMP2E
3607 011146 001407      BEQ      $231
  
```

|      |        |        |        |        |        |            |                               |
|------|--------|--------|--------|--------|--------|------------|-------------------------------|
| 3608 |        |        |        |        |        |            | :GO TO XCSR                   |
| 3609 | 011150 |        |        |        |        |            | LET TMP := TMP + #2           |
| 3610 | 011150 | 062737 | 000002 | 012370 | ADD    | #2,TMP     |                               |
| 3611 |        |        |        |        |        |            | :ENABLE XMIT INTERR           |
| 3612 | 011156 |        |        |        |        |            | LET @TMP := @TMP SET.BY #BIT6 |
| 3613 | 011156 | 052777 | 000100 | 00'204 | BIS    | #BIT6,@TMP |                               |
| 3614 | 011164 |        |        |        |        |            | ELSE                          |
| 3615 | 011164 | 000402 |        |        | BR     | \$232      |                               |
| 3616 | 011166 |        |        |        | \$231: |            |                               |
| 3617 |        |        |        |        |        |            | :ALL DONE                     |
| 3618 | 011166 |        |        |        |        |            | LET TMP4 :- TMP4 - #1         |
| 3619 | 011166 | 005337 | 012414 |        | DEC    | TMP4       |                               |
| 3620 | 011172 |        |        |        |        |            | ENDIF                         |
| 3621 | 011172 |        |        |        | \$232: |            |                               |
| 3622 | 011172 |        |        |        |        |            | ENDSRV                        |
| 3623 | 011172 | 000002 |        |        | RTI    |            |                               |
| 3624 |        |        |        |        |        |            | BGNSRV X2SRV                  |
| 3625 | 011174 |        |        |        | X2SRV: |            |                               |
| 3626 | 011174 |        |        |        |        |            | LET TMP :- RCSR               |
| 3627 | 011174 |        |        |        |        |            | LET TMP : TMP + #26           |
| 3628 | 011174 | 013737 | 001260 | 012370 | MOV    | RCSR,TMP   |                               |
| 3629 | 011202 |        |        |        |        |            | LET @TMP : TMP2               |
| 3630 | 011202 | 062737 | 000026 | 012370 | ADD    | #26,TMP    |                               |
| 3631 | 011210 |        |        |        |        |            | LET TMP2 :- TMP2 + #1         |
| 3632 | 011210 | 013777 | 012402 | 001152 | MOV    | TMP2,@TMP  |                               |
| 3633 | 011216 |        |        |        |        |            | :GO BACK TO XCSR              |
| 3634 | 011216 | 005237 | 012402 |        | INC    | TMP2       | LET TMP :- TMP - #2           |
| 3635 |        |        |        |        |        |            | :DISABLE XMIT INTERRUPTS,     |
| 3636 | 011222 |        |        |        |        |            | :NOT EXER DOUBLE BUFF.        |
| 3637 | 011222 | 162737 | 000002 | 012370 | SUB    | #2,TMP     | LET @TMP : @TMP CLR.BY #BIT6  |
| 3638 |        |        |        |        |        |            | ENDSRV                        |
| 3639 |        |        |        |        |        |            |                               |
| 3640 | 011230 |        |        |        |        |            |                               |
| 3641 | 011230 | 042777 | 000100 | 001132 | BIC    | #BIT6,@TMP |                               |
| 3642 | 011236 |        |        |        |        |            |                               |
| 3643 | 011236 | 000002 |        |        | RTI    |            |                               |
| 3644 |        |        |        |        |        |            | BGNSRV R3SRV                  |
| 3645 | 011240 |        |        |        | R3SRV: |            |                               |
| 3646 | 011240 |        |        |        |        |            | LET TMP : RCSR                |
| 3647 | 011240 |        |        |        |        |            | LET TMP : TMP + #32           |
| 3648 | 011240 | 013737 | 001260 | 012370 | MOV    | RCSR,TMP   |                               |
| 3649 | 011246 |        |        |        |        |            | LET (R3)+ : @TMP              |
| 3650 | 011246 | 062737 | 000032 | 012370 | ADD    | #32,TMP    |                               |
| 3651 | 011254 |        |        |        |        |            | IF TMP3 NE TMP3E THEN         |
| 3652 | 011254 | 017723 | 001110 |        | MOV    | @TMP,(R3)+ |                               |
| 3653 | 011260 |        |        |        |        |            |                               |
| 3654 | 011260 | 023737 | 012406 | 012410 | CMP    | TMP3,TMP3E |                               |
| 3655 | 011266 | 001407 |        |        | BEQ    | \$233      |                               |
| 3656 |        |        |        |        |        |            | :GO TO XCSR                   |
| 3657 | 011270 |        |        |        |        |            | LET TMP :- TMP + #2           |
| 3658 | 011270 | 062737 | 000002 | 012370 | ADD    | #2,TMP     |                               |
| 3659 |        |        |        |        |        |            | :ENABLE XMIT INTERR           |
| 3660 | 011276 |        |        |        |        |            | LET @TMP := @TMP SET.BY #BIT6 |
| 3661 | 011276 | 052777 | 000100 | 001064 | BIS    | #BIT6,@TMP |                               |
| 3662 | 011304 |        |        |        |        |            | ELSE                          |
| 3663 | 011304 | 000402 |        |        | BR     | \$234      |                               |

```

3664 011306          $233:
3665                                     ;ALL DONE
3666 011306          LET TMP4 : TMP4 - #1
3667 011306 005337 012414          DEC      TMP4
3668 011312          ENDIF
3669 011312          $234:
3670 011312          ENDSRV
3671 011312 000002          RTI
3672
3673 011314          BGNSRV X3SRV
3674 011314          X3SRV:
3675 011314          LET TMP :- RCSR
3676 011314 013737 001260 012370          MOV      RCSR,TMP
3677 011322          LET TMP :- TMP + #36
3678 011322 062737 000036 012370          ADD      #36,TMP
3679 011330          LET @TMP : TMP3
3680 011330 013777 012406 001032          MOV      TMP3,@TMP
3681 011336          LET TMP3 := TMP3 + #1
3682 011336 005237 012406          INC      TMP3
3683                                     ;GO BACK TO XCSR
3684 011342          LET TMP :- TMP - #2
3685 011342 162737 000002 012370          SUB      #2,TMP
3686                                     ;DISABLE XMIT INTERRUPTS,
3687                                     ;NOT EXER DOUBLE BUFF
3688 011350          LET @TMP :- @TMP CLR.BY #BIT6
3689 011350 042777 000100 001012          BIC      #BIT6,@TMP
3690 011356          ENDSRV
3691 011356 000002          RTI
3692 011360          SEREND:
  
```

3693  
 3694  
 3695  
 3696  
 3697  
 3698  
 3699  
 3700  
 3701  
 3702  
 3703  
 3704  
 3705  
 3706  
 3707  
 3708  
 3709  
 3710  
 3711  
 3712  
 3713  
 3714  
 3715  
 3716  
 3717  
 3718  
 3719  
 3720  
 3721  
 3722  
 3723  
 3724  
 3725  
 3726  
 3727  
 3728  
 3729  
 3730  
 3731  
 3732  
 3733  
 3734  
 3735

011360 000100  
 011560 000100  
 011760 000100  
 012160 000100  
 012360 000000  
 012362 000000  
 012364 000000  
 012366 000000  
 012370 000000  
 012372 000000  
 012374 000000  
 012376 000000  
 012400 000000  
 012402 000000  
 012404 000000  
 012406 000000  
 012410 000002  
 012414 000000  
 012416 011360  
 012420 011560  
 012422 011760  
 012424 012160  
 012426 000040  
 012430 000100  
 012432 000140  
 012434 000200  
 012436 000100  
 012440 000200  
 012442 000300  
 012444 000400

;\*\* RECEIVER STATUS TABLES \*\*

CH0TAB: .BLKW 100  
 CH1TAB: .BLKW 100  
 CH2TAB: .BLKW 100  
 CH3TAB: .BLKW 100  
 STATEND:  
 CHMSK: .WORD 0  
 CHCTR: .WORD 0  
 ACTCH: .WORD 0  
 ERWRD: .WORD 0

TMP: .WORD 0

MP0: 0  
 TMP0E: 0  
 TMP1: 0  
 TMP1E: 0  
 TMP2: 0  
 TMP2E: 0  
 TMP3: 0  
 TMP3E: .BLKW 2

TMP4: .WORD

TABL6: CH0TAB  
 CH1TAB  
 CH2TAB  
 CH3TAB

TABL7: 40  
 100  
 140  
 200

TABL8: 100  
 200  
 300  
 400

:7 BIT WDS 8 BIT WDS  
 : 0- 37 0- 77  
 : 40- 77 100-177  
 :100-137 200-277  
 :140-177 300-377

:MASK OF ACTIVE CHANNELS  
 :CHANNEL CTR  
 :ACTIVE CHANNEL CTR  
 :ERROR WORD, LOW BYTE IS NO. OF DATA ERRORS  
 :HIGH BYTE INDICATES ERROR FLAG DETECTED

:TEMP STORAGE FOR SERVICE ROUTINES

:START WORD FOR CH 0  
 :END WORD FOR CH 0  
 :START 1  
 :END 1  
 :START 2  
 :END 2  
 :START 3  
 :END 3

:7 BIT WORD TABLE

:8 BIT WORD TABLE

3736  
3737  
3738  
3739  
3740  
3741  
3742  
3743  
3744  
3745  
3746  
3747  
3748  
3749  
3750  
3751  
3752  
3753  
3754  
3755  
3756  
3757  
3758  
3759  
3760  
3761  
3762  
3763  
3764  
3765  
3766  
3767  
3768  
3769  
3770  
3771  
3772  
3773  
3774  
3775  
3776  
3777  
3778  
3779  
3780  
3781  
3782  
3783  
3784  
3785  
3786  
3787  
3788  
3789  
3790  
3791

```

*****
ROUTINE:TIMER
THIS ROUTINE IS USED TO TEST THE STATUS OF ANY BIT
IN ANY REGISTER.
INPUTS:
    HOWLONG      THE MAXIMUM AMOUNT OF TIME TO SPEND IN
                  THIS ROUTINE.
    WHICHBIT     A MASK WITH THE BIT(S) SET THAT ARE
                  TO BE CHECKED.
    REG          A POINTER TO THE REGISTER TO BE CHECKED
    SETCLR       THE DESIRED RESULTS
                  EITHER #SET OR #CLEAR
OUTPUT:
    THE 'C' BIT IS SET TO INDICATE AN ERROR
    BUT IT IS TESTED BY THE IF.ERROR STATEMENT

NOTE::  THE USE OF (R5) IS PART OF THE LINKAGE
        MECHANISM BETWEEN THE CALLER AND THE CALLED
*****

```

ROUTINE TIMER &lt;HOWLONG,WHICHBIT,REG,SETCLR&gt;

```

TIMER:
MOV    REG(R5),REGSAV
MOV    HOWLONG(R5),TIMSAV
CLRB   FLAG

;      START OF AN INFINITE LOOP
LOOP

$237:
;      ; TEST TO SEE IF WHICHBIT IS SET
;      IF      WHICHBIT(R5) NOTSETIN @REGSAV THEN
BIT     WHICHBIT(R5),@REGSAV
BNE     $241
CLRB    HOLDSC
BR      $242
$241:
;      LET      HOLDSC :B- #0
;      ELSE
;      LET      HOLDSC :B #SET ; REMEMBER THIS
MOV     #SET,HOLDSC
;      ENDIF
$242:
;      ; NOW SEE IF THAT WAS WHAT WE WANTED
;      IFB      HOLDSC EQ SETCLR(R5) THEN
CMP     HOLDSC,SETCLR(R5)
BNE     $2'3
;      ; JUST THE THING WE NEEDED

```

```

3792 012522          LET      FLAG :B= #TRUE
3793 012522 112737 000001 012612      MOVB      #TRUE,FLAG
3794 012530          ENDIF
3795 012530          $243:
3796
3797 012530          EXIFB    FLAG EQ #TRUE OR TIMSAV LE #0
3798 012530 123727 012612 000001      CMPB      FLAG,#TRUE
3799 012536 001414      BEQ      $240
3800 012540 005737 012610      TST      TIMSAV
3801 012544 003411      BLE      $240
3802
3803          ; ONE WAY OR THE OTHER, WE ARE DONE
3804          ; IF WE ARE STILL HERE THEN HANG AROUND A WHILE
3805          WAITMS 1          ;WAIT FOR 1 MILLI-SECONDS
3806 012546 010546      MOV      R5,-(SP)
3807 012550 012745      MOV      #1,-(R5)
3808 012554 004737 012620      JSR      PC,WAIT
3809 012560 012605      MOV      (SP)+,R5
3810 012562          LET      TIMSAV : TIMSAV - #1 ; COUNTING DOWN
3811 012562 005337 012610      DEC      TIMSAV
3812 012566          ENDLOOP          ; CONTINUED AT THE TOP
3813 012566 000737      BR      $237
3814 012570          $240:
3815
3816          ; ONLY 2 WAYS TO GET HERE
3817          ; 1). WE RAN OUT OF TIME---ERROR .
3818          ; 2). THE BIT IS IN THE CORRECT CONDITION--GOOD .
3819
3820 012570          IFB      FLAG EQ #TRUE THEN
3821 012570 123727 012612 000001      CMPB      FLAG,#TRUE
3822 012576 001001      BNE      $244
3823 012600          RETURN NO.ERROR      ; GOOD
3824 012600 000405      BR      $235
3825 012602          ENDIF
3826 012602          $244:
3827 012602          RETURN ERROR      ; BAD
3828 012602 000261      SEC
3829 012604 000404      BR      $236
3830
3831 012606 000000      REGSAV: .WORD 0
3832 012610 000000      TIMSAV: .WORD 0
3833 012612 000      FLAG: .BYTE 0
3834 012613 000      HOLDSC: .BYTE 0
3835
3836          ; WE ARE DONE GO BACK HOME
3837          ENDRTN
3838 012614 000241      $235:
3839 012616          $236:
3840 012616 000207      CLC
                      RTS      PC
  
```

```

3841
3842
3843
3844
3845
3846
3847
3848
3849
3850 012620
3851 012620
3852 012620
3853 012626
3854 012626 016501 000000
3855 012632
3856 012632 012702 000001
3857 012636 000402
3858 012640
3859 012640 062702 000001
3860 012644
3861 012644 020201
3862 012646 101010
3863 012650
3864 012650 005003
3865 012652 000401
3866 012654
3867 012654 005203
3868 012656
3869 012656 020327 000100
3870 012662 003001
3871 012664
3872 012664 000773
3873 012666
3874 012666
3875 012666 000764
3876 012670
3877 012670
3878 012676
3879 012676
3880 012676
3881 012676 000207

*****
* ROUTINE:WAIT
* THIS ROUTINE IS USED TO DELAY EXECUTION OF THE
* MAIN PROGRAM FOR A SPECIFIED AMOUNT OF TIME.
* THIS IS ACCOMPLISHED BY INCREMENTING A
* REGISTER UP TO A LIMIT. THE INNER LOOP IS SET
* TO APPROXIMATE 1 MILLI SEC.
*****
ROUTINE WAIT <TIME>
WAIT:
      PUSH <R1,R2,R3>
      LET R1 := TIME(R5)
      INCRU R2 FROM #1 TO R1 BY #1
      MOV TIME(R5),R1
      MOV #1,R2
      BR $247
$250: ADD #01,R2
$247: CMP R2,R1
      BHI $251
      INCR R3 FROM #0 TO #100 BY #1
      CLR R3
      BR $252
$253: INC R3
$252: CMP R3,#100
      BGT $254
      ENDINC
$254: BR $253
      ENDINC
$251: BR $250
      POP <R3,R2,R1>
      ENDRTN
$245:
$246: RTS PC
  
```

```

3882
3883
3884 .SBTTL ROUTINE TO SIZE THE BUS
3885
3886 012700 005737 001252 SIZE: TST $DEVM ;BR IF INITIALLY SET
3887 012704 001060 BNE 4$ ;BYPASS AUTO-SIZING
3888
3889 012706 012737 013610 000004 1$: MOV #INTSRV,ERRVEC
3890 012714 012737 000340 000006 MOV #PR7,ERRVEC+2
3891
3892 012722 013737 001250 013514 MOV $BASE,TMP6
3893 012730 012737 000001 013512 MOV #1,SFTREG
3894
3895 012736 005037 013616 2$: CLR INTFLAG
3896 012742 005777 000546 TST @TMP6 ;SEE IF THERE
3897 012746 005737 013616 TST INTFLAG ;GOT INTERRUPT?
3898 012752 001003 BNE 3$ ;BR IF YES (NO DEVICE)
3899 012754 053737 013512 001252 BIS SFTREG,$DEVM ;ELSE THERE, SET MAP
3900
3901 012762 062737 000010 013514 3$: ADD #10,TMP6 ;GO TO NEXT CHANNEL
3902 012770 006337 013512 ASL SFTREG
3903 012774 001360 BNE 2$ ;BR IF MORE TO GO
3904
3905 012776 012737 000006 000004 MOV #6,ERRVEC ;RESTORE
3906 013004 005037 000006 CLR ERRVEC+2
3907
3908 013010 005737 001252 TST $DEVM ;ANYONE THERE?
3909 013014 001014 BNE 4$ ;BR IF YES
3910 013016 TYPTXT <<CRLF>!NO DEV TO TEST!> ;LOOP HERE TILL PROBLEM FIXED
3911 013044 000777 BR . ;& RESTART AT 200
3912
3913 013046 105737 001221 4$: TSTB $USWR+1 ;SEE IF ANY CONSOLE
3914 013052 001434 BEQ 6$ ;BR IF NONE
3915
3916 013054 010046 MOV R0,-(SP) ;SAVE R0
3917 013056 113700 001221 MOVB $USWR+1,R0 ;GET CONSOLE DEVICE X2
3918 013062 020027 000010 CMP R0,#10
3919 013066 003005 BGT 5$ ;BR IF TOO HI
3920 013070 056037 013516 001252 BIS TABL1-2(R0),$DEVM ;SET CONSOLE BIT ON MAP
3921 013076 012600 MOV (SP)+,R0 ;RESTORE R0
3922 013100 000421 BR 6$
3923
3924 013102 5$: TYPTXT <<CRLF>!CONSOLE IN $USWR WRONG!>
3925 013140 000137 014362 JMP $EOP ;START ALL OVER
3926 ;STACK WILL RESET IN 'LOOP'
3927
3927 013144 013737 001252 013502 6$: MOV $DEVM,MASK1
3928 013152 013737 001252 013504 MOV $DEVM,MASK2
3929 013160 013737 001252 013506 MOV $DEVM,MASK3
3930 013166 013737 001252 013510 MOV $DEVM,MASK4
3931
3932 013174 042737 177760 013502 BIC #^C<17>,MASK1 ;SAVE ONLY MODULE 1 CHANNELS
3933 013202 042737 177417 013504 BIC #^C<360>,MASK2 ;
3934 013210 042737 170377 013506 BIC #^C<7400>,MASK3 ;
3935 013216 042737 007777 013510 BIC #^C<170000>,MASK4 ;
3936
3937 013224 006237 013504 ASR MASK2
  
```

```

3938 013230 006237 013504      ASR      MASK2
3939 013234 006237 013504      ASR      MASK2
3940 013240 006237 013504      ASR      MASK2      ;RIGHT JUSTIFY MODULE 2 CHANNELS
3941
3942 013244 000337 013506      SWAB     MASK3      ;
3943
3944 013250 000337 013510      SWAB     MASK4
3945 013254 006237 013510      ASR      MASK4
3946 013260 006237 013510      ASR      MASK4
3947 013264 006237 013510      ASR      MASK4
3948 013270 006237 013510      ASR      MASK4      ;
3949
3950 013274      TYPTXT    <<CRLF>.WILL TEST:.<CRLF>>
3951
3952 013320 012700 013502      MOV      #MASK1,RO
3953 013324 012737 000061 013474      MOV      #61,MODNUM      ;INIT MOD #
3954 013332 012737 000060 013500      MOV      #60,CHNUM+2      ;INIT CHANNEL #
3955
3956 013340 012037 013516      7$:      MOV      (RO)+,MSK
3957 013344 001001      BNE      8$      ;BR IF ANY THERE
3958 013346 000441      BR       11$      ;ELSE TRY NEXT MODULE
3959
3960 013350      8$:      TYPTXT    <<CRLF>.MODULE .>
3961 013370 104401 013474      TYPE     ,MODNUM
3962 013374      TYPTXT    <. CHANNEL .>
3963
3964 013416 032737 000001 013516 9$:      BIT      #BIT0,MSK
3965 013424 001402      BEQ      10$      ;BR IF NOT TESTING THAT CHANNEL
3966 013426 104401 013476      TYPE     ,CHNUM      ;ELSE TYPE IT
3967
3968 013432 005237 013500      10$:     INC      CHNUM+2      ;BUMP CHANNEL CTR
3969 013436 006237 013516      ASR      MSK
3970 013442 001365      BNE      9$      ;BR IF MORE TO CHECK
3971 013444 012737 000060 013500      MOV      #60,CHNUM+2      ;ELSE RESET CH #
3972
3973 013452 005237 013474      11$:     INC      MODNUM      ;BUMP MOD #
3974 013456 023727 013474 000065      CMP      MODNUM,#65      ;ALL DONE?
3975 013464 001325      BNE      7$      ;BR IF NO
3976 013466 104401 001171      TYPE     ,CRLF
3977 013472 000207      RTS      PC      ;ELSE EXIT
3978
3979 013474 000061      MODNUM: .ASCIZ  /1/
3980 013476 020040 000060      CHNUM: .ASCIZ  / 0/
3981
3982 013502 000000      MASK1:  0      ;CONTAINS MODULE 1 CHANNELS
3983 013504 000000      MASK2:  0      ;
3984 013506 000000      MASK3:  0      ;
3985 013510 000000      MASK4:  0      ;
3986 013512 000000      SFTREG: 0
3987 013514 000000      TMP6:   0
3988 013516 000000      MSK:    0
3989 013520 000010      TABL1:  10      ;SET $DEVM IF CONSOLE ON MODULE 1
3990 013522 000200      ;
3991 013524 004000      ;
3992 013526 100000      ;
3993

```

```

3994
3995
3996
3997      .SBTTL  ROUTINE TO TEST FOR CONSOLE DEVICE
3998
3999 013530 113737 001221 013606 TSTCON: MOVB  $USWR+1,TMP7      ;GET CONSOLE INFO ONLY
4000 013536 006237 013606      ASR      TMP7              ;RIGHT JUSTIFY
4001 013542 023737 013606 001206      CMP      TMP7,$UNIT      ;SEE IF UNIT UNDER TEST HAS CONSOLE
4002 013550 001403      BEQ      1$              ;BR IF YES
4003 013552 005037 013604      CLR      CONSOLE          ;ELSE CLEAR STALE INFO
4004 013556 000207      RTS      PC
4005
4006 013560 013737 001144 014156 1$:  MOV      $TKS,ADDRESS      ;FORCE TTY ADDRESS
4007 013566 012737 000060 014160      MOV      #TKVEC,VECTOR      ;      VECTOR
4008 013574 012737 000001 013604      MOV      #TRUE,CONSOLE      ;SET FLAG
4009 013602 000207      RTS      PC
4010
4011 013604 000000      CONSOLE:      0              ;FLAG, 1= TESTING CONSOLE DEVICE
4012 013606 000000      TMP7:      0              ;HOLD CONSOLE #
4013
4014
4015
4016      .SBTTL  INTSRV  INTERRUPT SERVICE ROUTINE
4017
4018      ;*****
4019      ;*      THIS GLOBAL ROUTINE DOES NOTHING BUT INCREMENT
4020      ;*      'INTFLAG' EACH TIME IT IS CALLED.  IT ASSUMES
4021      ;*      THAT THE MAIN CALLING ROUTINE WILL  KNOW  WHAT
4022      ;*      TO LOOK FOR.
4023      ;*****
4024 013610      BGNSRV INTSRV      ;GLOBAL INTERRUPT SERVICE ROUTINE
4025 013610      INTSRV:
4026      ;ADD 1 TO 'INTERRUPT OCCURED' FLAG
4027 013610      ;      LET INTFLAG := INTFLAG + #1
4028 013610 005237 013616      INC      INTFLAG
4029 013614      ENDSRV      ;THAT'S ALL
4030 013614 000002      RTI
4031 013616 000000      INTFLAG:  0
4032
  
```

```

4033
4034 .SBTTL ROUTINE TO SET UP FOR NEXT MODULE/CHANNEL ADDRESS
4035
4036
4037 *****
4038 THIS ROUTINE CAUSES ADRS TO POINT TO THE
4039 ADDRESS OF DLV11-J UNDER TEST, ADRS +2 TO
4040 POINT TO THE VECTOR OF THE DLV11-J UNDER TEST.
4041 *****
4042
4043 CYCLE: MOV $BASE,ADDRESS
4044 MOV $VECT1,VECTOR
4045
4046 CMP $UNIT,#5 ;DONE ALL MODULES?
4047 BNE 1$ ;BR IF NO
4048 CLR PHASE2 ;RE-INIT
4049 CLR CHAN
4050 MOV #1,$UNIT
4051 CLR P1CNT
4052 JMP $EOP ;& EXIT BACK TO 'LOOP'
4053 ;STACK WILL GET RESET THERE.
4054 1$: MOV $UNIT,R0
4055 ASL R0 ;MULT BY 2 TO INDEX INTO TABLES
4056 ADD TABL2-2(R0),ADDRESS ;ADD MODULE OFFSET
4057 ADD TABL2-2(R0),VECTOR ;ADD VECTOR OFFSET
4058 MOV @TABL3-2(R0),MASK ;STORE CHANNELS IN 'MASK'
4059
4060 CLR CONSOLE
4061 TST MASK ;ANY CHANNELS TO BE TESTED?
4062 BNE 2$ ;BR IF YES
4063 INC $UNIT ;ELSE BUMP UNIT #
4064 JMP CYCLE ;TRY AGAIN
4065
4066 2$: CMP PHASE2,#TRUE ;DOING PHASE 2?
4067 BEQ 6$ ;BR IF YES
4068 CMP CHAN,#4 ;ELSE, DONE ALL CHANNELS?
4069 BNE 4$ ;BR IF NO
4070
4071 CLR CHAN ;ELSE BACK TO 0
4072 BIT #WRAP,$USWR ;DOING DATA WRAP TESTS?
4073 BNE 3$ ;BR IF YES
4074 INC $UNIT ;ELSE GO TO NEXT UNIT, NO PHASE 2
4075 JMP CYCLE ;& DO IT
4076
4077 3$: MOV #TRUE,PHASE2
4078 CLR P1CNT
4079 JMP CYCLE ;GO TO PHASE 2 TESTING
4080
4081 4$: MOV CHAN,R0
4082 ASL R0 ;MULT BY 2 TO INDEX TABLE
4083 ADD TABL4(R0),ADDRESS ;ADD CHANNEL OFFSET
4084 ADD TABL4(R0),VECTOR ;ADD VECTOR OFFSET
4085 BIT TABL5(R0),MASK ;IS THAT CHANNEL TO BE TESTED?
4086 BEQ 7$ ;BR IF NO
4087
4088 CMP CHAN,#3 ;IS IT CHAN 3?

```

D 8  
CVDLAAO DLV11-J TEST MACY11 30G(1063) 17-NOV-78 09:39 PAGE 95  
CVDLAB.P11 17-NOV-78 09:38 ROUTINE TO SET UP FOR NEXT MODULE/CHANNEL ADDRESS

SEQ 0094

```
4089 014066 001002          BNE 5$          ;BR IF NO
4090 014070 004737 013530    JSR PC,TSTCON    ;ELSE SEE IF ITS THE CONSOLE
4091                                     ;BUMP CHANNEL CTR FOR NEXT TIME
4092 014074 005237 014164    5$: INC CHAN      ;GO TO PHASE 1 TESTS
4093 014100 012701 014156    6$: MOV #ADDRESS,ADRS
4094 014104 000207          RTS PC
4095                                     ;TRY AGAIN
4096 014106 005237 014164    7$: INC CHAN
4097 014112 000137 013620    JMP CYCLE
4098
4099 TABL2: 0          ;OFFSET FOR MODULE 1
4100         40        ;          2
4101         100       ;          3
4102         140       ;          4
4103
4104 TABL3: MASK1      ;MODULE 1 CHANNEL DATA
4105         MASK2     ;          2
4106         MASK3     ;          3
4107         MASK4     ;          4
4108
4109 TABL4: 0          ;CHANNEL 0 OFFSET
4110         10        ;          1
4111         20        ;          2
4112         30        ;          3
4113
4114 TABL5: BIT0       ;BIT TO TEST CHANNEL 0 PRESENT
4115         BIT1      ;          1
4116         BIT2      ;          2
4117         BIT3      ;          3
4118
4119 ADDRESS: 0
4120 VECTOR: 0
4121 MASK: 0
4122 CHAN: 0
4123 PHASE2: 0
4124 P1CNT: 0          ;CTR TO PRINT 'PHASE 1' ONLY ONCE
4125
4126
```

```
4127
4128 014172
4129 014172
4130 014172
4131 014212
4132 014220
4133 014240 113737 001114 001176
4134 014246
4135 014256
4136 014274
4137 014302
4138 014320
4139 014326
4140 014346
4141 014354 104401 001171
4142 014360
4143 014360
4144 014360
4145 014360 000207
4146
```

MYTYPE:

```
TYPTXT <<CRLF>*TEST # *>
TYPOCT $TESTN
TYPTXT <*,ERROR # *>
MOVB $ITEMB,$FATAL ; APT FATAL ERROR NUMBER
TYPOCS $FATAL
TYPTXT <*,PC = *>
TYPOCT $ERRPC
TYPTXT <*,CSR: *>
TYPOCT DLADD
TYPTXT <*,VECTOR: *>
TYPOCT DLVEC
TYPE , $CRLF
```

ENDRTN

\$255:  
\$256:

RTS PC

```

4147 .SBTTL END OF PASS ROUTINE
4148
4149 ;*****
4150 ;*INCREMENT THE PASS NUMBER ($PASS)
4151 ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
4152 ;*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
4153 ;*IF THERES A MONITOR GO TO IT
4154 ;*IF THERE ISN'T JUMP TO LOOP
4155
4156 $EOP:
4157 014362 SCOPE
4158 014362 CLR $TSTNM ;;ZERO THE TEST NUMBER
4159 014364 005037 001102 CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
4160 014370 005037 001160 INC $PASS ;;INCREMENT THE PASS NUMBER
4161 014374 005237 001202 BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
4162 014400 042737 100000 001202 DEC (PC)+ ;;LOOP?
4163 014410 000001 $EOPCT: .WORD 1
4164 014412 003022 BGT $DOAGN ;;YES
4165 014414 012737 MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
4166 014416 000001 $ENDCT: .WORD 1
4167 014420 014410 $EOPCT
4168 014422 104401 014467 TYPE ,SENDMG ;;TYPE 'END PASS #'
4169 014426 013746 001202 MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
4170 014432 104405 TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
4171 014434 104401 014464 TYPE ,SENULL ;;TYPE A NULL CHARACTER
4172 014440 013700 000042 $GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
4173 014444 001405 BEQ $DOAGN ;;BRANCH IF NO MONITOR
4174 014446 000005 RESET ;;CLEAR THE WORLD
4175 014450 004710 $ENDAD: JSR PC,(R0) ;;GO TO MONITOR
4176 014452 000240 NOP ;;SAVE ROOM
4177 014454 000240 NOP ;;FOR
4178 014456 000240 NOP ;;ACT11
4179 014460 $DOAGN:
4180 014460 000137 JMP @ (PC)+ ;;RETURN
4181 014462 001732 $RTNAD: .WORD LOOP
4182 014464 377 377 000 $ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
4183 014467 015 042412 042116 $ENDMG: .ASCII <15><12>/END PASS #/
4184 014474 050040 051501 020123
4185 014502 000043
  
```

```

4186 .SBTTL TYPE ROUTINE
4187
4188 *****
4189 *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
4190 *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
4191 *NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
4192 *NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
4193 *NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
4194
4195 *CALL:
4196 *1) USING A TRAP INSTRUCTION
4197 * TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
4198 *OR
4199 * TYPE
4200 * MESADR
4201 *
4202
4203 014504 105737 001157 $TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
4204 014510 100002 BPL 1$ ;;BR IF YES
4205 014512 000000 HALT ;;HALT HERE IF NO TERMINAL
4206 014514 000430 BR 3$ ;;LEAVE
4207 014516 010046 1$: MOV R0,-(SP) ;;SAVE R0
4208 014520 017600 000002 MOV @2(SP),R0 ;;GET ADDRESS OF ASCIZ STRING
4209 014524 122737 000001 001214 CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
4210 014532 001011 BNE 62$ ;;NO,GO CHECK FOR APT CONSOLE
4211 014534 132737 000100 001215 BITB #APTPOOL,$ENVM ;;SPOOL MESSAGE TO APT
4212 014542 001405 BEQ 62$ ;;NO,GO CHECK FOR CONSOLE
4213 014544 010037 014554 MOV R0,61$ ;;SETUP MESSAGE ADDRESS FOR APT
4214 014550 004737 015550 JSR PC,$ATY3 ;;SPOOL MESSAGE TO APT
4215 014554 000000 61$: .WORD 0 ;;MESSAGE ADDRESS
4216 014556 132737 000040 001215 62$: BITB #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
4217 014564 001003 BNE 60$ ;;YES,SKIP TYPE OUT
4218 014566 112046 2$: MOVB (R0)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
4219 014570 001005 BNE 4$ ;;BR IF IT ISN'T THE TERMINATOR
4220 014572 005726 TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
4221 014574 012600 60$: MOV (SP)+,R0 ;;RESTORE R0
4222 014576 062716 000002 3$: ADD #2,(SP) ;;ADJUST RETURN PC
4223 014602 000002 RTI ;;RETURN
4224 014604 122716 000011 4$: CMPB #HT,(SP) ;;BRANCH IF <HT>
4225 014610 001430 BEQ 8$
4226 014612 122716 000200 CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
4227 014616 001006 BNE 5$
4228 014620 005726 TST (SP)+ ;;POP <CR><LF> EQUIV
4229 014622 104401 TYPE ;;TYPE A CR AND LF
4230 014624 001171 $CRLF
4231 014626 105037 014762 CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
4232 014632 000755 BR 2$ ;;GET NEXT CHARACTER
4233 014634 004737 014716 5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
4234 014640 123726 001156 6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
4235 014644 001350 BNE 2$ ;;IF NO GO GET NEXT CHAR.
4236 014646 013746 001154 MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
4237 AND THE NULL CHAR.
4238 014652 105366 000001 7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
4239 014656 002770 BLT 6$ ;;BR IF NO--GO POP THE NULL OFF OF STACK
4240 014660 004737 014716 JSR PC,$TYPEC ;;GO TYPE A NULL
4241 014664 105337 014762 DECB $CHARCNT ;;DO NOT COUNT AS A COUNT

```

```

4242 014670 000770          BR      7$          ;;LOOP
4243
4244          ;HORIZONTAL TAB PROCESSOR
4245
4246 014672 112716 000040      8$:      MOVB      #' , (SP)          ;;REPLACE TAB WITH SPACE
4247 014676 004737 014716      9$:      JSR      PC,$TYPEC          ;;TYPE A SPACE
4248 014702 132737 000007 014762      BITB      #7,$CHARCNT          ;;BRANCH IF NOT AT
4249 014710 001372          BNE      9$          ;;TAB STOP
4250 014712 005726          TST      (SP)+          ;;POP SPACE OFF STACK
4251 014714 000724          BR      2$          ;;GET NEXT CHARACTER
4252 014716 105777 164226      $TYPEC: TSTB      @ $TPS          ;;WAIT UNTIL PRINTER IS READY
4253 014722 100375          BPL      $TYPEC
4254 014724 116677 000002 164220      MOVB      2(SP),@ $TPB          ;;LOAD CHAR TO BE TYPED INTO DATA REG.
4255 014732 122766 000015 000002      CMPB      #CR,2(SP)          ;;IS CHARACTER A CARRIAGE RETURN?
4256 014740 001003          BNE      1$          ;;BRANCH IF NO
4257 014742 105037 014762      CLRB      $CHARCNT          ;;YES--CLEAR CHARACTER COUNT
4258 014746 000406          BR      $TYPEX          ;;EXIT
4259 014750 122766 000012 000002 1$:      CMPB      #LF,2(SP)          ;;IS CHARACTER A LINE FEED?
4260 014756 001402          BEQ      $TYPEX          ;;BRANCH IF YES
4261 014760 105227          INCB      (PC)+          ;;COUNT THE CHARACTER
4262 014762 000000          $CHARCNT: .WORD 0          ;;CHARACTER COUNT STORAGE
4263 014764 000207          $TYPEX: RTS      PC
4264

```

```

4265 .SBTTL TTY INPUT ROUTINE
4266
4267 *****
4268 .ENABL LSB
4269
4270 *****
4271 *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
4272 *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
4273 *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
4274 *WHEN OPERATING IN TTY F-AG MODE.
4275 014766 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
4276 014774 001074 BNE 15$ ;;BRANCH IF NO
4277 014776 105777 164142 TSTB @STKS ;;CHAR THERE?
4278 015002 100071 BPL 15$ ;;IF NO, DON'T WAIT AROUND
4279 015004 117746 164136 MOVB @STKB, -(SP) ;;SAVE THE CHAR
4280 015010 042716 177600 BIC #^C177, (SP) ;;STRIP-OFF THE ASCII
4281 015014 022726 000007 CMP #7, (SP)+ ;;IS IT A CONTROL G?
4282 015020 001062 BNE 15$ ;;NO, RETURN TO USER
4283 015022 123727 001134 000001 CMPB $AJTOB, #1 ;;ARE WE RUNNING IN AUTO-MODE?
4284 015030 001456 BEQ 15$ ;;BRANCH IF YES
4285
4286 015032 104401 015513 $GTSWR: TYPE .SCNTLG ;;ECHO THE CONTROL-G (^G)
4287 015036 104401 015520 TYPE .SMSWR ;;TYPE CURRENT CONTENTS
4288 015042 013746 000176 MOV SWREG, -(SP) ;;SAVE SWREG FOR TYPEOUT
4289 015046 104402 TYPDC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
4290 015050 104401 015531 TYPE .SMNEW ;;PROMPT FOR NEW SWR
4291 015054 005046 19$: CLR -(SP) ;;CLEAR COUNTER
4292 015056 005046 CLR -(SP) ;;THE NEW SWR
4293 015060 105777 164060 7$: TSTB @STKS ;;CHAR THERE?
4294 015064 100375 BPL 7$ ;;IF NOT TRY AGAIN
4295
4296 015066 117746 164054 MOVB @STKB, -(SP) ;;PICK UP CHAR
4297 015072 042716 177600 BIC #^C177, (SP) ;;MAKE IT 7-BIT ASCII
4298
4299
4300
4301 015076 021627 000025 9$: CMP (SP), #25 ;;IS IT A CONTROL-U?
4302 015102 001005 BNE 10$ ;;BRANCH IF NOT
4303 015104 104401 015506 TYPE .SCNTLU ;;YES, ECHO CONTROL-U (^U)
4304 015110 062706 000006 20$: ADD #6, SP ;;IGNORE PREVIOUS INPUT
4305 015114 000757 BR 19$ ;;LET'S TRY IT AGAIN
4306
4307
4308 015116 021627 000015 10$: CMP (SP), #15 ;;IS IT A <CR>?
4309 015122 001022 BNE 16$ ;;BRANCH IF NO
4310 015124 005766 000004 TST 4(SP) ;;YES, IS IT THE FIRST CHAR?
4311 015130 001403 BEQ 11$ ;;BRANCH IF YES
4312 015132 016677 000002 164000 MOV 2(SP), @SWR ;;SAVE NEW SWR
4313 015140 062706 000006 11$: ADD #6, SP ;;CLEAR UP STACK
4314 015144 104401 001171 14$: TYPE .SCRLF ;;ECHO <CR> AND <LF>
4315 015150 123727 001135 000001 CMPB $INTAG, #1 ;;RE-ENABLE TTY KBD INTERRUPTS?
4316 015156 001003 BNE 15$ ;;BRANCH IF NOT
4317 015160 012777 000100 163756 MOV #100, @STKS ;;RE-ENABLE TTY KBD INTERRUPTS
4318 015166 000002 R'I ;;RETURN
4319 015170 004737 014716 16$: JSR PL, $TYPEC ;;ECHO CHAR
4320 015174 021627 000060 CMP (SP), #60 ;;CHAR < 0?

```

```

4321 015200 002420          BLT      18$          ;;BRANCH IF YES
4322 015202 021627 000067    CMP      (SP),#67      ;;CHAR > 7?
4323 015206 003015          BGT      18$          ;;BRANCH IF YES
4324 015210 042726 000060    BIC      #60,(SP)+    ;;STRIP-OFF ASCII
4325 015214 005766 000002    TST      2(SP)        ;;IS THIS THE FIRST CHAR
4326 015220 001403          BEQ      17$          ;;BRANCH IF YES
4327 015222 006316          ASL      (SP)          ;;NO, SHIFT PRESENT
4328 015224 006316          ASL      (SP)          ;;CHAR OVER TO MAKE
4329 015226 006316          ASL      (SP)          ;;ROOM FOR NEW ONE.
4330 015230 005266 000002    17$: INC      2(SP)        ;;KEEP COUNT OF CHAR
4331 015234 056616 177776    BIS      -2(SP),(SP)    ;;SET IN NEW CHAR
4332 015240 000707          BR       7$           ;;GET THE NEXT ONE
4333 015242 104401 001170    18$: TYPE    $QUES      ;;TYPE ?<CR><LF>
4334 015246 000720          BR       20$          ;;SIMULATE CONTROL-U
4335          .DSABL   LSB
4336
4337
4338          ;*****
4339          ;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4340          ;CALL:
4341          ;RDCHR          ;;INPUT A SINGLE CHARACTER FROM THE TTY
4342          ;RETURN HERE   ;;CHARACTER IS ON THE STACK
4343          ;              ;;WITH PARITY BIT STRIPPED OFF
4344          ;
4345
4346 015250 011646          $RDCHR: MOV      (SP),-(SP)    ;;PUSH DOWN THE PC
4347 015252 016666 000004 000002    MOV      4(SP),2(SP)    ;;SAVE THE PS
4348 015260 105777 163660    1$: TSTB     @STKS          ;;WAIT FOR
4349 015264 100375          BPL      1$           ;;A CHARACTER
4350 015266 117766 163654 000004    MOVB    @STKB,4(SP)    ;;READ THE TTY
4351 015274 042766 177600 000004    BIC      #^C<177>,4(SP)    ;;GET RID OF JUNK IF ANY
4352 015302 026627 000004 000023    CMP      4(SP),#23    ;;IS IT A CONTROL-S?
4353 015310 001013          BNE      3$           ;;BRANCH IF NO
4354 015312 105777 163626    2$: TSTB     @STKS          ;;WAIT FOR A CHARACTER
4355 015316 100375          BPL      2$           ;;LOOP UNTIL ITS THERE
4356 015320 117746 163622    MOVB    @STKB,-(SP)    ;;GET CHARACTER
4357 015324 042716 177600    BIC      #^C177,(SP)    ;;MAKE IT 7-BIT ASCII
4358 015330 022627 000021    CMP      (SP)+,#21    ;;IS IT A CONTROL-Q?
4359 015334 001366          BNE      2$           ;;IF NOT DISCARD IT
4360 015336 000750          BR       1$           ;;YES, RESUME
4361 015340 026627 000004 000140    3$: CMP      4(SP),#140    ;;IS IT UPPER CASE?
4362 015346 002407          BLT      4$           ;;BRANCH IF YES
4363 015350 026627 000004 000175    CMP      4(SP),#175    ;;IS IT A SPECIAL CHAR?
4364 015356 003003          BGT      4$           ;;BRANCH IF YES
4365 015360 042766 000040 000004    BIC      #40,4(SP)    ;;MAKE IT UPPER CASE
4366 015366 000002          4$: RTI              ;;GO BACK TO USER
4367          ;*****
4368          ;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
4369          ;CALL:
4370          ;RDLIN          ;;INPUT A STRING FROM THE TTY
4371          ;RETURN HERE   ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4372          ;              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
4373          ;
4374 015370 010346          $RDLIN: MOV      R3,-(SP)    ;;SAVE R3
4375 015372 012703 015476    1$: MOV      #2,TTYIN,R3    ;;GET ADDRESS
4376 015376 022703 015506    2$: CM      #TTYIN+8.,R3    ;;BUFFER FULL?

```

|      |        |        |        |        |                  |                |                                           |
|------|--------|--------|--------|--------|------------------|----------------|-------------------------------------------|
| 4377 | 015402 | 101405 |        |        | BLOS             | 4\$            | ::BR IF YES                               |
| 4378 | 015404 | 104410 |        |        | RDCHR            |                | ::GO READ ONE CHARACTER FROM THE TTY      |
| 4379 | 015406 | 112613 |        |        | MOVB             | (SP)+,(R3)     | ::GET CHARACTER                           |
| 4380 | 015410 | 122713 | 000177 | 10\$:  | CMPB             | #177,(R3)      | ::IS IT A RUBOUT                          |
| 4381 | 015414 | 001003 |        |        | BNE              | 3\$            | ::SKIP IF NOT                             |
| 4382 | 015416 | 104401 | 001170 | 4\$:   | TYPE             | ,\$QUES        | ::TYPE A '?'                              |
| 4383 | 015422 | 000763 |        |        | BR               | 1\$            | ::CLEAR THE BUFFER AND LOOP               |
| 4384 | 015424 | 111337 | 015474 | 3\$:   | MOVB             | (R3),9\$       | ::ECHO THE CHARACTER                      |
| 4385 | 015430 | 104401 | 015474 |        | TYPE             | ,\$            |                                           |
| 4386 | 015434 | 122723 | 000015 |        | CMPB             | #15,(R3)+      | ::CHECK FOR RETURN                        |
| 4387 | 015440 | 001356 |        |        | BNE              | 2\$            | ::LOOP IF NOT RETURN                      |
| 4388 | 015442 | 105063 | 177777 |        | CLRB             | -1(R3)         | ::CLEAR RETURN (THE 15)                   |
| 4389 | 015446 | 104401 | 001172 |        | TYPE             | ,\$LF          | ::TYPE A LINE FEED                        |
| 4390 | 015452 | 012603 |        |        | MOV              | (SP)+,R3       | ::RESTORE R3                              |
| 4391 | 015454 | 011646 |        |        | MOV              | (SP),-(SP)     | ::ADJUST THE STACK AND PUT ADDRESS OF THE |
| 4392 | 015456 | 016666 | 000004 | 000002 | MOV              | 4(SP),2(SP)    | ::FIRST ASCII CHARACTER ON IT             |
| 4393 | 015464 | 012766 | 015476 | 000004 | MOV              | #\$TTYIN,4(SP) |                                           |
| 4394 | 015472 | 000002 |        |        | RTI              |                | ::RETURN                                  |
| 4395 | 015474 | 000    |        | 9\$:   | .BYTE            | 0              | ::STORAGE FOR ASCII CHAR. TO TYPE         |
| 4396 | 015475 | 000    |        |        | .BYTE            | 0              | ::TERMINATOR                              |
| 4397 | 015476 | 000010 |        |        | \$TTYIN: .BLKB   | 8.             | ::RESERVE 8 BYTES FOR TTY INPUT           |
| 4398 | 015506 | 052536 | 005015 | 000    | \$CNTLU: .ASCIIZ | /'^U/<15><12>  | ::CONTROL 'U'                             |
| 4399 | 015513 | 136    | 006507 | 000012 | \$CNTLG: .ASCIIZ | /'^G/<15><12>  | ::CONTROL 'G'                             |
| 4400 | 015520 | 005015 | 053523 | 020122 | \$MSWR: .ASCIIZ  | <15><12>/SWR   | /                                         |
| 4401 | 015526 | 020075 | 000    |        |                  |                |                                           |
| 4402 | 015531 | 040    | 047040 | 053505 | \$MNEW: .ASCIIZ  | / NEW = /      |                                           |
| 4403 | 015536 | 036440 | 000040 |        |                  |                |                                           |

```

4404 .SBTTL APT COMMUNICATIONS ROUTINE
4405
4406 *****
4407 015542 112737 000001 016006 $ATY1: MOVB #1,$FFLG ;;TO REPORT FATAL ERROR
4408 015550 112737 000001 016004 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE
4409 015556 000403 BR $ATYC
4410 015560 112737 000001 016006 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
4411 015566 $ATYC:
4412 015566 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
4413 015570 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
4414 015572 105737 016004 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
4415 015576 001450 BEQ 5$ ;;IF NOT: BR
4416 015600 122737 000001 001214 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
4417 015606 001031 BNE 3$ ;;IF NOT: BR
4418 015610 132737 000100 001215 BITB #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
4419 015616 001425 BEQ 3$ ;;IF NOT: BR
4420 015620 017600 000004 MOV @4(SP),R0 ;;GET MESSAGE ADDR.
4421 015624 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
4422 015632 005737 001174 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
4423 015636 001375 BNE 1$ ;;IF NOT: WAIT
4424 015640 010037 001210 MOV R0,$MSGAD ;;PUT ADDR IN MAILBOX
4425 015644 105720 2$: TSTB (R0)+ ;;FIND END OF MESSAGE
4426 015646 001376 BNE 2$
4427 015650 163700 001210 SUB $MSGAD,R0 ;;SUB START OF MESSAGE
4428 015654 006200 ASR R0 ;;GET MESSAGE LGTH IN WORDS
4429 015656 010037 001212 MOV R0,$MSGGLT ;;PUT LENGTH IN MAILBOX
4430 015662 012737 000004 001174 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
4431 015670 000413 BR 5$
4432 015672 017637 000004 015716 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
4433 015700 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
4434 015706 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
4435 015712 004737 014504 JSR PC,$TYPE ;;CALL TYPE MACRO
4436 015716 000000 4$: .WORD 0
4437 015720 5$:
4438 015720 105737 016006 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
4439 015724 001416 BEQ 12$ ;;IF NOT: BR
4440 015726 005737 001214 TST $ENV ;;RUNNING UNDER APT?
4441 015732 001413 BEQ 12$ ;;IF NOT: BR
4442 015734 005737 001174 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
4443 015740 001375 BNE 11$ ;;IF NOT: WAIT
4444 015742 017637 000004 001176 MOV @4(SP),$FATAL ;;GET ERROR #
4445 015750 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
4446 015756 005237 001174 INC $MSGTYPE ;;TELL APT TO TAKE ERROR
4447 015762 105037 016006 12$: CLRB $FFLG ;;CLEAR FATAL FLAG
4448 015766 105037 016005 CLRB $LFLG ;;CLEAR LOG FLAG
4449 015772 105037 016004 CLRB $MFLG ;;CLEAR MESSAGE FLAG
4450 015776 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
4451 016000 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
4452 016002 000207 RTS PC ;;RETURN
4453 016004 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
4454 016005 000 $LFLG: .BYTE 0 ;;LOG FLAG
4455 016006 000 $FFLG: .BYTE 0 ;;FATAL FLAG
4456 016010 .EVEN
4457 000200 APTSIZE=200
4458 000001 APTENV=001
4459 000100 APTPOOL=100

```

CVDLAA0 DLV11-J TEST MACY11 30G(1063) 17-NOV-78 09:39 PAGE 104<sup>M 8</sup>  
CVDLAB.P11 17-NOV-78 09:38 APT COMMUNICATIONS ROUTINE

SEQ 0103

4460

000040

APTC SUP=040

```

4461 .SBTTL ERROR HANDLER ROUTINE
4462
4463 *****
4464 *THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
4465 *SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
4466 *AND GO TO MYTYPE ON ERROR
4467 *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4468 *SW15=1 HALT ON ERROR
4469 *SW13=1 INHIBIT ERROR TYPEOUTS
4470 *SW10=1 BELL ON ERROR
4471 *SW09=1 LOOP ON ERROR
4472 *CALL
4473 *
4474 ERROR N ;;ERROR EMT AND N=ERROR ITEM NUMBER
4475 $ERROR:
4476 016010 104407 001103 163114 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
4477 016012 105237 001103 163104 'S: INCB $ERFLG ;;SET THE ERROR FLAG
4478 016016 001775 BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
4479 016020 013777 001102 163114 MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
4480 016026 032777 002000 163104 BIT #BIT10,@SWR ;;BELL ON ERROR?
4481 016034 001402 BEQ 1$ ;;NO - SKIP
4482 016036 104401 001164 TYPE $BELL ;;RING BELL
4483 016042 005237 001112 1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
4484 016046 011637 001116 MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
4485 016052 162737 000002 001116 SUB #2,$ERRPC
4486 016060 117737 163032 001114 MOVB @ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
4487 016066 032777 020000 163044 BIT #BIT13,@SWR ;;SKIP TYPEOUT IF SET
4488 016074 001004 BNE 20$ ;;SKIP TYPEOUTS
4489 016076 004737 014172 JSR PC,MYTYPE ;;GO TO USER ERROR ROUTINE
4490 016102 104401 001171 TYPE $CRLF
4491 016106 122737 000001 001214 20$: CMPB #APT.LV,$ENV ;;RUNNING IN APT MODE
4492 016114 001007 BNE 2$ ;;NO,SKIP APT ERROR REPORT
4493 016116 113737 001114 016130 MOVB $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER
4494 016124 004737 015560 JSR PC,$ATY4 ;;REPORT FATAL ERROR TO APT
4495 016130 000 .BYTE 0
4496 016131 000 .BYTE 0
4497 016132 000777 163000 21$: BR 22$ ;;APT ERROR LOOP
4498 016134 005777 2$: TST @SWR ;;HALT ON ERROR
4499 016140 100002 BPL 3$ ;;SKIP IF CONTINUE
4500 016142 000000 HALT ;;HALT ON ERROR!
4501 016144 104407 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
4502 016146 032777 001000 162764 3$: BIT #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?
4503 016154 001402 BEQ 4$ ;;BR IF NO
4504 016156 013716 001110 MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
4505 016162 005737 001162 4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
4506 016166 001402 BEQ 5$ ;;BR IF NONE
4507 016170 013716 001162 MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
4508 016174 022737 014450 000042 5$: CMP #SENDAD,@#4? ;;ACT-11 AUTO-ACCEPT?
4509 016174 001001 BNE 6$ ;;BRANCH IF NO
4510 016202 000000 HALT ;;YES
4511 016204 000002 6$: RTI .RETURN
4512
4513
4514

```

```
4515 .SBTTL SCOPE HANDLER ROUTINE
4516
4517 *****
4518 *THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
4519 *AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
4520 *AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
4521 *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4522 *SW14=1 LOOP ON TEST
4523 *SW11=1 INHIBIT ITERATIONS
4524 *SW09=1 LOOP ON ERROR
4525 *SW08=1 LOOP ON TEST IN SWR<7:0>
4526 *CALI
4527 * SCOPE ;:SCOPE-IOT
4528
4529 $SCOPE:
4530 016210 104407 040000 162720 1$: CKSWR ;:TEST FOR CHANGE IN SOFT-SWR
4531 016212 032777 040000 162720 BIT #BIT14,@SWR ;:LOOP ON PRESENT TEST?
4532 016220 001114 BNE $OVER ;:YES IF SW14=1
4533 ;:*****START OF CODE FOR THE XOR TESTER*****
4534 016222 000416 $XTSTR: BR 6$ ;:IF RUNNING ON THE 'XOR' TESTER CHANGE
4535 ;:THIS INSTRUCTION TO A 'NOP' (NOP=240)
4536 016224 013746 000004 MOV @ERRVEC,-(SP) ;:SAVE THE CONTENTS OF THE ERROR VECTOR
4537 016230 012737 016250 000004 MOV #5,$@ERRVEC ;:SET FOR TIMEOUT
4538 016236 005737 177060 TST @177060 ;:TIME OUT ON XOR?
4539 016242 012637 000004 MOV (SP)+,@ERRVEC ;:RESTORE THE ERROR VECTOR
4540 016246 000463 BR $SVLAD ;:GO TO THE NEXT TEST
4541 016250 022626 5$: CMP (SP)+,(SP)+ ;:CLEAR THE STACK AFTER A TIME OUT
4542 016252 012637 000004 MOV (SP)+,@ERRVEC ;:RESTORE THE ERROR VECTOR
4543 016256 000423 BR 7$ ;:LOOP ON THE PRESENT TEST
4544 016260 6$: ;:*****END OF CODE FOR THE XOR TESTER*****
4545 016260 032777 000400 162652 BIT #BIT08,@SWR ;:LOOP ON SPEC. TEST?
4546 016266 001404 BEQ 2$ ;:BR IF NO
4547 016270 127737 162644 001102 CMPB @SWR,$TSTNM ;:ON THE RIGHT TEST? SWR<7:0>
4548 016276 001465 BEQ $OVER ;:BR IF YES
4549 016300 105737 001103 2$: TSTB $ERFLG ;:HAS AN ERROR OCCURRED?
4550 016304 001421 BEQ 3$ ;:BR IF NO
4551 016306 123737 001115 001103 CMPB $ERMAX,$ERFLG ;:MAX. ERRORS FOR THIS TEST OCCURRED?
4552 016314 101115 BHI 3$ ;:BR IF NO
4553 016316 032777 001000 162614 BIT #BIT09,@SWR ;:LOOP ON ERROR?
4554 016324 001404 BEQ 4$ ;:BR IF NO
4555 016326 013737 001110 001106 7$: MOV $LPERR,$LPADR ;:SET LOOP ADDRESS TO LAST SCOPE
4556 016334 000446 BR $OVER
4557 016336 105037 001103 4$: CLRB $ERFLG ;:ZERO THE ERROR FLAG
4558 016342 005037 001160 CLR $TIMES ;:CLEAR THE NUMBER OF ITERATIONS TO MAKE
4559 016346 000415 BR 1$ ;:ESCAPE TO THE NEXT TEST
4560 016350 032777 004000 162562 3$: BIT #BIT11,@SWR ;:INHIBIT ITERATIONS?
4561 016356 001011 BNE 1$ ;:BR IF YES
4562 016360 005737 001202 TST $PASS ;:IF FIRST PASS OF PROGRAM
4563 016364 001406 BEQ 1$ ;:INHIBIT ITERATIONS
4564 016366 005237 001104 INC $ICNT ;:INCREMENT ITERATION COUNT
4565 016372 023737 001160 001104 CMP $TIMES,$ICNT ;:CHECK THE NUMBER OF ITERATIONS MADE
4566 016400 002024 BGE $OVER ;:BR IF MORE ITERATION REQUIRED
4567 016402 012737 000001 001104 1$: MOV #1,$ICNT ;:REINITIALIZE THE ITERATION COUNTER
4568 016410 013737 016466 001160 MOV $MXCNT,$TIMES ;:SET NUMBER OF ITERATIONS TO DO
4569 016416 105237 001102 $SVLAD: INCB $TSTNM ;:COUNT TEST NUMBERS
4570 016422 113737 001102 001200 MOVB $TSTNM,$TESTN ;:SET TEST NUMBER IN APT MAILBOX
```

CVDLAAO DLV11-J TEST MACY11 30G(1063) 17-NOV-78 09:39 PAGE 107<sup>9</sup>  
CVDLAB.P11 17-NOV-78 09:38 SCOPE HANDLER ROUTINE

SEQ 0106

|      |        |        |        |        |                |                   |                                         |
|------|--------|--------|--------|--------|----------------|-------------------|-----------------------------------------|
| 4571 | 016430 | 011637 | 001106 |        | MOV            | (SP), \$LPADR     | :::SAVE SCOPE LOOP ADDRESS              |
| 4572 | 016434 | 011637 | 001110 |        | MOV            | (SP), \$LPERR     | :::SAVE ERROR LOOP ADDRESS              |
| 4573 | 016440 | 005037 | 001162 |        | CLR            | \$ESCAPE          | :::CLEAR THE ESCAPE FROM ERROR ADDRESS  |
| 4574 | 016444 | 112737 | 000001 | 001115 | MOVB           | #1, \$ERMAX       | :::ONLY ALLOW ONE(1) ERROR ON NEXT TEST |
| 4575 | 016452 | 013777 | 001102 | 162462 | \$OVER: MOV    | \$TSTNM, @DISPLAY | :::DISPLAY TEST NUMBER                  |
| 4576 | 016460 | 013716 | 001106 |        | MOV            | \$LPADR, (SP)     | :::FUDGE RETURN ADDRESS                 |
| 4577 | 016464 | 000002 |        |        | RTI            |                   | :::FIXES PS                             |
| 4578 | 016466 | 003720 |        |        | \$MXCNT: 2000. |                   | :::MAX. NUMBER OF ITERATIONS            |

```

4579 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
4580
4581 *****
4582 *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
4583 *SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
4584 *NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
4585 *BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
4586 *REPLACED WITH SPACES.
4587 *CALL:
4588 *      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
4589 *      TYPDS    ;;GO TO THE ROUTINE
4590
4591 $TYPDS:
4592 016470 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
4593 016472 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
4594 016474 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
4595 016476 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
4596 016500 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
4597 016502 012746 020200 MOV      #20200,-(SP)      ;;SET BLANK SWITCH AND SIGN
4598 016506 016605 000020 MOV      20(SP),R5      ;;GET THE INPUT NUMBER
4599 016512 100004      BPL      1$          ;;BR IF INPUT IS POS.
4600 016514 005405      NEG      R5          ;;MAKE THE BINARY NUMBER POS.
4601 016516 112766 000055 000001 1$:      MOVB     #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
4602 016524 005000      CLR      R0          ;;ZERO THE CONSTANTS INDEX
4603 016526 012703 016704      MOV      #SDBLK,R3      ;;SETUP THE OUTPUT POINTER
4604 016532 112723 000040      MOVB     #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
4605 016536 005002      CLR      R2          ;;CLEAR THE BCD NUMBER
4606 016540 016001 016674      MOV      $DTBL(R0),R1      ;;GET THE CONSTANT
4607 016544 1601C5      3$:      SUB      R1,R5      ;;FORM THIS BCD DIGIT
4608 016546 002402      BLT      4$          ;;BR IF DONE
4609 016550 005202      INC      R2          ;;INCREASE THE BCD DIGIT BY 1
4610 016552 000774      BR       3$
4611 016554 060105      4$:      ADD      R1,R5      ;;ADD BACK THE CONSTANT
4612 016556 005702      TST      R2          ;;CHECK IF BCD DIGIT 0
4613 016560 001002      BNE      5$          ;;FALL THROUGH IF 0
4614 016562 105716      TSTB     (SP)          ;;STILL DOING LEADING 0'S?
4615 016564 100407      BMI      7$          ;;BR IF YES
4616 016566 106316      5$:      ASLB      (SP)          ;;MSD?
4617 016570 103003      BCC      6$          ;;BR IF NO
4618 016572 116663 000001 177777      MOVB     1(SP),-1(R3)      ;;YES--SET THE SIGN
4619 016600 052702 000060      BIS      #'0,R2      ;;MAKE THE BCD DIGIT ASCII
4620 016604 052702 000040      6$:      BIS      #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
4621 016610 110223      MOVB     R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
4622 016612 005720      TST      (R0)+      ;;JUST INCREMENTING
4623 016614 020027 000010      CMP      R0,#10      ;;CHECK THE TABLE INDEX
4624 016620 002746      BLT      2$          ;;GO DO THE NEXT DIGIT
4625 016622 003002      BGT      8$          ;;GO TO EXIT
4626 016624 010502      MOV      R5,R2      ;;GET THE LSD
4627 016626 000764      BR       6$          ;;GO CHANGE TO ASCII
4628 016630 105726      8$:      TSTB     (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
4629 016632 100003      BPL      9$          ;;BR IF NO
4630 016634 116663 177777 177776      MOVB     -1(SP),-2(R3)      ;;YES--SET THE SIGN FOR TYPING
4631 016642 105013      9$:      CLRB      (R3)          ;;SET THE TERMINATOR
4632 016644 012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
4633 016646 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
4634 016650 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2

```

CVDLAAO DLV11-J TEST MACY11 30G(1063) 17-NOV-78 09:39 PAGE 109  
CVDLAB.P11 17-NOV-78 09:38 CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

SEQ 0108

```
4635 016652 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
4636 016654 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
4637 016656 104401 016704 TYPE      $DBLK      ;;NOW TYPE THE NUMBER
4638 016662 016666 000002 000004 MOV      2(SP),4(SP)  ;;ADJUST THE STACK
4639 016670 012616      MOV      (SP)+,(SP)
4640 016672 000002      RTI                      ;;RETURN TO USER
4641 016674 023420      $DTBL: 10000.
4642 016676 001750      1000.
4643 016700 000144      100.
4644 016702 000012      10.
4645 016704 000004      $DBLK: .BLKW 4
```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

:*****
:THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
:OCTAL (ASCII) NUMBER AND TYPE IT.
:$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
:CALL:
:      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
:      TYPOS      ;;CALL FOR TYPEOUT
:      .BYTE     N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
:      .BYTE     M      ;;M=1 OR 0
:                        ;;1=TYPE LEADING ZEROS
:                        ;;0=SUPPRESS LEADING ZEROS
:
:$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
:$TYPOS OR $TYPOC
:CALL:
:      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
:      TYPON      ;;CALL FOR TYPEOUT
:
:$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
:CALL:
:      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
:      TYPOC      ;;CALL FOR TYPEOUT
:
$TYPOS: MOV      @ (SP),-(SP)      ;;PICKUP THE MODE
:      MOV      1(SP), $OFILL      ;;LOAD ZERO FILL SWITCH
:      MOV      (SP)+, $OMODE+1      ;;NUMBER OF DIGITS TO TYPE
:      ADD      #2, (SP)      ;;ADJUST RETURN ADDRESS
:      BR      $TYPON
$TYPOC: MOV      #1, $OFILL      ;;SET THE ZERO FILL SWITCH
:      MOV      #6, $OMODE+1      ;;SET FOR SIX(6) DIGITS
$TYPON: MOV      #5, $OCNT      ;;SET THE ITERATION COUNT
:      MOV      R3,-(SP)      ;;SAVE R3
:      MOV      R4,-(SP)      ;;SAVE R4
:      MOV      R5,-(SP)      ;;SAVE R5
:      MOV      $OMODE+1, R4      ;;GET THE NUMBER OF DIGITS TO TYPE
:      NEG      R4
:      ADD      #6, R4      ;;SUBTRACT IT FOR MAX. ALLOWED
:      MOV      R4, $OMODE      ;;SAVE IT FOR USE
:      MOV      $OFILL, R4      ;;GET THE ZERO FILL SWITCH
:      MOV      12(SP), R5      ;;PICKUP THE INPUT NUMBER
:      CLR      R3      ;;CLEAR THE OUTPUT WORD
1$: ROL      R5      ;;ROTATE MSB INTO 'C'
:      BR      3$      ;;GO DO MSB
2$: ROL      R5      ;;FORM THIS DIGIT
:      ROL      R5
:      ROL      R5
:      MOV      R5, R3
3$: ROL      R3      ;;GET LSB OF THIS DIGIT
:      DECB     $OMODE      ;;TYPE THIS DIGIT?
:      BPL      7$      ;;BR IF NO
:      BIC      #177770, R3      ;;GET RID OF JUNK
:      BNE      4$      ;;TEST FOR 0
:      TST      R4      ;;SUPPRESS THIS 0?
:      BEQ      5$      ;;BR IF YES

```

|      |        |        |               |          |       |             |                                   |
|------|--------|--------|---------------|----------|-------|-------------|-----------------------------------|
| 4702 | 017056 | 005204 |               | 4\$:     | INC   | R4          | ::DON'T SUPPRESS ANYMORE 0'S      |
| 4703 | 017060 | 052703 | 000060        |          | BIS   | #'0,R3      | ::MAKE THIS DIGIT ASCII           |
| 4704 | 017064 | 052703 | 000040        | 5\$:     | BIS   | #',R3       | ::MAKE ASCII IF NOT ALREADY       |
| 4705 | 017070 | 110337 | 017134        |          | MOVB  | R3,8\$      | ::SAVE FOR TYPING                 |
| 4706 | 017074 | 104401 | 017134        |          | TYPE  | ,8\$        | ::GO TYPE THIS DIGIT              |
| 4707 | 017100 | 105337 | 017136        | 7\$:     | DECB  | \$OCNT      | ::COUNT BY 1                      |
| 4708 | 017104 | 003347 |               |          | BGT   | 2\$         | ::BR IF MORE TO DO                |
| 4709 | 017106 | 002402 |               |          | BLT   | 6\$         | ::BR IF DONE                      |
| 4710 | 017110 | 005204 |               |          | INC   | R4          | ::INSURE LAST DIGIT ISN'T A BLANK |
| 4711 | 017112 | 000744 |               |          | BR    | 2\$         | ::GO DO THE LAST DIGIT            |
| 4712 | 017114 | 012605 |               | 6\$:     | MOV   | (SP)+,R5    | ::RESTORE R5                      |
| 4713 | 017116 | 012604 |               |          | MOV   | (SP)+,R4    | ::RESTORE R4                      |
| 4714 | 017120 | 012603 |               |          | MOV   | (SP)+,R3    | ::RESTORE R3                      |
| 4715 | 017122 | 016666 | 000002 000004 |          | MOV   | 2(SP),4(SP) | ::SET THE STACK FOR RETURNING     |
| 4716 | 017130 | 012616 |               |          | MOV   | (SP)+,(SP)  |                                   |
| 4717 | 017132 | 000002 |               |          | RTI   |             | ::RETURN                          |
| 4718 | 017134 | 000    |               | 8\$:     | .BYTE | 0           | ::STORAGE FOR ASCII DIGIT         |
| 4719 | 017135 | 000    |               |          | .BYTE | 0           | ::TERMINATOR FOR TYPE ROUTINE     |
| 4720 | 017136 | 000    |               | \$OCNT:  | .BYTE | 0           | ::OCTAL DIGIT COUNTER             |
| 4721 | 017137 | 000    |               | \$OFILL: | .BYTE | 0           | ::ZERO FILL SWITCH                |
| 4722 | 017140 | 000000 |               | \$UMODE: | .WORD | 0           | ::NUMBER OF DIGITS TO TYPE        |

```

4723          .SBTTL TRAP DECODER
4724          1
4725          ;*****
4726          ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
4727          ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
4728          ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
4729          ;*GO TO THAT ROUTINE.
4730
4731          $TRAP: MOV      R0,-(SP)          ;;SAVE R0
4732          MOV      2(SP),R0              ;;GET TRAP ADDRESS
4733          IST      -(R0)                  ;;BACKUP BY 2
4734          MOVB     (R0),R0                ;;GET RIGHT BYTE OF TRAP
4735          ASL      R0                     ;;POSITION FOR INDEXING
4736          MOV      $TRPAD(R0),R0          ;;INDEX TO TABLE
4737          RTS      R0                     ;;GO TO ROUTINE
4738
4739          ;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO
4740
4741          $TRAP2: MOV      (SP),-(SP)       ;;MOVE THE PC DOWN
4742          MOV      4(SP),2(SP)             ;;MOVE THE PSW DOWN
4743          RTI                               ;;RESTORE THE PSW
4744
4745          .SBTTL TRAP TABLE
4746
4747          ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
4748          ;*BY THE 'TRAP' INSTRUCTION.
4749
4750          :          ROUTINE
4751          :          -----
4752          $TRPAD: .WORD      $TRAP2
4753          $TYPE      ;;CALL=TYPE          TRAP+1(104401)  TTY TYPEOUT ROUTINE
4754          $TYPOC     ;;CALL=TYPOC         TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
4755          $TYPOS     ;;CALL=TYPOS         TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
4756          $TYPON     ;;CALL=TYPON        TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
4757          $TYPDS     ;;CALL=TYPDS        TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
4758
4759          $GTSWR     ;;CALL=GTSWR         TRAP+6(104406)  GET SOFT-SWR SETTING
4760
4761          $CKSWR     ;;CALL=CKSWR         TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
4762          $RDCHR     ;;CALL=RDCHR         TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
4763          $RDLIN     ;;CALL=RDLIN        TRAP+11(104411) TTY TYPEIN STRING ROUTINE
4764
4765          ;MAXMEM FOR APT = 17400
4766          ;MAXMEM NON APT 17500 (ABS LOADER)
4767
4768          .END
  
```

|                 |                 |                 |                 |                 |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| ABASE = 176500  | AECT1 = 000300  | DLVEC = 001256  | P1CNT = 014170  | SW04 = 000020   |
| ACDW1 = 000000  | AECT2 = 000000  | DSWR = 177570   | RBUF = 001262   | SW05 = 000040   |
| ACDW2 = 000000  | BIT0 = 000001   | EMTVFC = 000030 | RCH0 = 007724   | SW06 = 000100   |
| ACPUOP = 000000 | BIT00 = 000001  | ERRCHK = 006500 | RCH1 = 007726   | SW07 = 000200   |
| ACTCH = 012364  | BIT01 = 000002  | ERRCNT = 005622 | RCH2 = 007730   | SW08 = 000400   |
| ADDRS = 014156  | BIT02 = 000004  | ERROR = 100000  | RCH3 = 007732   | SW09 = 001000   |
| ADDW0 = 000000  | BIT03 = 000010  | ERRORF = 003304 | RCSR = 001260   | SW1 = 000002    |
| ADDW1 = 000000  | BIT04 = 000020  | ERRVEC = 000004 | RCVRD0 = 000200 | SW10 = 002000   |
| ADDW10 = 000000 | BIT05 = 000040  | ERWRD = 012366  | RCVRIE = 000100 | SW11 = 004000   |
| ADDW11 = 000000 | BIT06 = 000100  | EVENOD = 000004 | RCVR0S = 007424 | SW12 = 010000   |
| ADDW12 = 000000 | BIT07 = 000200  | EXITFL = 003306 | RCVR1S = 007504 | SW13 = 020000   |
| ADDW13 = 000000 | BIT08 = 000400  | FLAG = 012612   | RCVR2S = 007564 | SW14 = 040000   |
| ADDW14 = 000000 | BIT09 = 001000  | FRERR = 020000  | RCVR3S = 007644 | SW15 = 100000   |
| ADDW15 = 000000 | BIT1 = 000002   | GTSWR = 104406  | RDATA0 = 000001 | SW2 = 000004    |
| ADDW2 = 000000  | BIT10 = 002000  | HOLDSC = 012613 | RDATA1 = 000002 | SW3 = 000010    |
| ADDW3 = 000000  | BIT11 = 004000  | HOWLON = 000000 | RDATA2 = 000004 | SW4 = 000020    |
| ADDW4 = 000000  | BIT12 = 010000  | HT = 000011     | RDATA3 = 000010 | SW5 = 000040    |
| ADDW5 = 000000  | BIT13 = 020000  | I = 001270      | RDATA4 = 000020 | SW6 = 000100    |
| ADDW6 = 000000  | BIT14 = 040000  | INTFLA = 013616 | RDATA5 = 000040 | SW7 = 000200    |
| ADDW7 = 000000  | BIT15 = 100000  | INTRTA = 007744 | RDATA6 = 000100 | SW8 = 000400    |
| ADDW8 = 000000  | BIT2 = 000004   | INTSRV = 013610 | RDATA7 = 000200 | SW9 = 001000    |
| ADDW9 = 000000  | BIT3 = 000010   | IOTVEC = 000020 | RDCHR = 104410  | TABEND = 007764 |
| ADEVCT = 000000 | BIT4 = 000020   | LF = 000012     | RDLIN = 104411  | TABL1 = 013520  |
| ADEVN = 000000  | BIT5 = 000040   | LOOP = 001732   | RDRUN = 000001  | TABL2 = 014116  |
| ADRS = 0000001  | BIT6 = 000100   | MASK = 014162   | REC = 005664    | TABL3 = 014126  |
| AENV = 000000   | BIT7 = 000200   | MASK1 = 013502  | REG = 000004    | TABL4 = 014136  |
| AENVN = 000000  | BIT8 = 000400   | MASK2 = 013504  | REGSAV = 012606 | TABL5 = 014146  |
| AFATAL = 000000 | BIT9 = 001000   | MASK3 = 013506  | RESVEC = 000010 | TABL6 = 012416  |
| AMADR1 = 000000 | BPTVEC = 000014 | MASK4 = 013510  | RHLD = 005770   | TABL7 = 012426  |
| AMADR2 = 000000 | BREAK = 000001  | MODNUM = 013474 | R0SRV = 010660  | TABL8 = 012436  |
| AMADR3 = 000000 | BRK = 000010    | MODTST = 007000 | R1SRV = 011000  | TBITVE = 000014 |
| AMADR4 = 000000 | CHAN = 014164   | MSK = 013516    | R2SRV = 011120  | TBUF = 001266   |
| AMAMS1 = 000000 | CHCNT = 007766  | MYTYPE = 014172 | R3SRV = 011240  | TCH0 = 007734   |
| AMAMS2 = 000000 | CHCTR = 012362  | NUMBER = 005624 | R5STAC = 001332 | TCH1 = 007736   |
| AMAMS3 = 000000 | CHMASK = 007764 | ORERR = 040000  | R6 = 0000006    | TCH2 = 007740   |
| AMAMS4 = 000000 | CHMSK = 012360  | PARITY = 000002 | R7 = 0000007    | TCH3 = 007742   |
| AMSGAD = 000000 | CHNUM = 013476  | PASS = 003302   | SEREND = 011360 | TCNT = 007772   |
| AMSGLG = 000000 | CHOTAB = 011360 | PERR = 010000   | SET = 177777    | TCSR = 001264   |
| AMSGTY = 000000 | CH1TAB = 011560 | PHASE2 = 014166 | SETCLR = 000006 | TDATA0 = 000001 |
| AMTYP1 = 000000 | CH2TAB = 011760 | PIRG = 177772   | SFTREG = 013512 | TDATA1 = 000002 |
| AMTYP2 = 000000 | CH3TAB = 012160 | PIRQVE = 000240 | SIZE = 012700   | TDATA2 = 000004 |
| AMTYP3 = 000000 | CKSWR = 104407  | PRIERR = 007770 | SRVEND = 007724 | TDATA3 = 000010 |
| AMTYP4 = 000000 | CLR = 000000    | PR0 = 000000    | STACK = 001100  | TDATA4 = 000020 |
| APASS = 000000  | CONSOL = 013604 | PR1 = 000040    | START = 001334  | TDATA5 = 000040 |
| APRIOR = 000000 | CR = 000015     | PR2 = 000100    | STATEN = 012360 | TDATA6 = 000100 |
| APTCU = 000040  | CRLF = 000200   | PR3 = 000140    | STKLMT = 177774 | TDATA7 = 000200 |
| APTSIZ = 000200 | CYCLE = 013620  | PR4 = 000200    | SWR = 001140    | TEMP = 007774   |
| APTSPO = 000100 | DATA = 005662   | PR5 = 000240    | SWREG = 000176  | TIME = 000000   |
| ASWREG = 000000 | DDISP = 177570  | PR6 = 000300    | SW0 = 000001    | TIMER = 012446  |
| ATESTN = 000000 | DISPLA = 001142 | PR7 = 000340    | SW00 = 000001   | TIMSAV = 012610 |
| AUNIT = 000000  | DISPRE = 000174 | PS = 177776     | SW01 = 000002   | TKVEC = 000060  |
| AUSWR = 001031  | DLADD = 001254  | PSW = 177776    | SW02 = 000004   | TMP = 012370    |
|                 | DLADDR = 176500 | PWRVEC = 000024 | SW03 = 000010   | TMP0 = 012372   |

|         |        |           |        |            |        |          |        |           |        |
|---------|--------|-----------|--------|------------|--------|----------|--------|-----------|--------|
| TMP0E   | 012374 | \$APTHD   | 001000 | \$F\$INC-  | 000210 | \$MXCNT  | 016466 | \$TYPEC   | 014716 |
| TMP1    | 012376 | \$ATYC    | 015566 | \$F\$LOO=  | 000200 | \$NESTL= | 177777 | \$TYPEX   | 014764 |
| TMP1E   | 012400 | \$ATY1    | 015542 | \$F\$NAM=  | 000160 | \$NSKO = | 000300 | \$TYPOC   | 016740 |
| TMP2    | 012402 | \$ATY3    | 015550 | \$F\$NO -  | 000403 | \$NSK1 = | 000210 | \$TYPON   | 016754 |
| TMP2E   | 012404 | \$ATY4    | 015560 | \$F\$OR -  | 000320 | \$NSK2 = | 000210 | \$TYPOS   | 016714 |
| TMP3    | 012406 | \$AUTOB   | 001134 | \$F\$RTN=  | 000300 | \$NSK3 = | 000110 | \$UNIT    | 001206 |
| TMP3E   | 012410 | \$BASE    | 001250 | \$F\$SFL=  | 000140 | \$NULL   | 001154 | \$UNITM   | 001010 |
| TMP4    | 012414 | \$BDADR   | 001122 | \$F\$SUNT= | 000130 | \$NWTST= | 000001 | \$USWR    | 001220 |
| TMP6    | 013514 | \$BDDAT   | 001126 | \$F\$WHI=  | 000120 | \$OCNT   | 017136 | \$VECT1   | 001244 |
| TMP7    | 013606 | \$BELL    | 001164 | \$F\$YES=  | 000402 | \$OMODE  | 017140 | \$VECT2   | 001246 |
| TPVEC = | 000064 | \$BGNLE=  | 177777 | \$GDADR    | 001120 | \$OVER   | 016452 | \$XTSTR   | 016222 |
| TRAN    | 005626 | \$CHARC   | 014762 | \$GDDAT    | 001124 | \$PASS   | 001202 | \$YESNO=  | 000001 |
| TRAPVE= | 000034 | \$CKSWR   | 014766 | \$GET42    | 014440 | \$PASTM  | 001006 | \$SARGC=  | 000000 |
| TRTVEC= | 000014 | \$CMTAG   | 001100 | \$GTSWR    | 015036 | \$QUES   | 001170 | \$S\$YTE= | 000402 |
| TRUE =  | 000001 | \$CM3 -   | 000000 | \$HD =     | 000000 | \$RDCHR  | 015250 | \$FLAG=   | 000001 |
| TSTCON  | 013530 | \$CNTLG   | 015513 | \$HIBTS    | 001000 | \$RDLIN  | 015370 | \$FROM=   | 000000 |
| TST1    | 002066 | \$CNTL    | 015506 | \$ICNT     | 001104 | \$RDSZ - | 000010 | \$GET4=   | 000000 |
| TST10   | 003440 | \$CPUOP   | 001222 | \$IFLEV=   | 177777 | \$RTNAD  | 014462 | \$LOC =   | 012576 |
| TST11   | 003566 | \$CRLF    | 001171 | \$INTAG    | 001135 | \$SAVLE= | 177777 | \$LOCN=   | 000000 |
| TST12   | 003712 | \$DBLK    | 016704 | \$ISK0 =   | 000001 | \$SCOPE  | 016210 | \$RETN=   | 000000 |
| TST13   | 004222 | \$DEVCT   | 001204 | \$ISK1 -   | 000001 | \$SETUP= | 000127 | \$RTN1=   | 000255 |
| TST14   | 004512 | \$DEVM    | 001252 | \$ISK2 -   | 000001 | \$SSKC = | 000253 | \$RTN2=   | 000256 |
| TST15   | 005046 | \$DOAGN   | 014460 | \$ITEMB    | 001114 | \$STUP - | 177777 | \$STO =   | 000001 |
| TST16   | 005322 | \$DTBL    | 016674 | \$LF       | 001172 | \$SVLAD  | 016416 | \$OFILL   | 017137 |
| TST17   | 005772 | \$ENDAD   | 014450 | \$LFLG     | 016005 | \$SVPC = | 000430 | \$1       | 002062 |
| TST2    | 002264 | \$ENDCT   | 014416 | \$LOCTA=   | 177777 | \$SWR =  | 167400 | \$10      | 002410 |
| TST20   | 006502 | \$ENDMG   | 014467 | \$LPADR    | 001106 | \$SWREG  | 001216 | \$100     | 005162 |
| TST21   | 007000 | \$ENULL   | 014464 | \$LPERR    | 001110 | \$SWRMK= | 000000 | \$101     | 005224 |
| TST22   | 007776 | \$ENV     | 001214 | \$LSTCN=   | 177777 | \$TAGLE= | 177777 | \$102     | 005240 |
| TST3    | 002440 | \$ENVM    | 001215 | \$LSTIN=   | 000000 | \$TAGNU= | 000257 | \$103     | 005276 |
| TST4    | 002614 | \$EOP     | 014362 | \$LSTST=   | 177777 | \$TEMP = | 000300 | \$104     | 005270 |
| TST5    | 002756 | \$EOPCT   | 014410 | \$LSTTA=   | 000000 | \$TESTN  | 001200 | \$105     | 005266 |
| TST6    | 003016 | \$ERFLG   | 001103 | \$MADR1    | 001226 | \$TIMES  | 001160 | \$106     | 005274 |
| TST7    | 003310 | \$ERMAX   | 001115 | \$MADR2    | 001232 | \$TKB    | 001146 | \$107     | 005320 |
| TYPDS = | 104405 | \$ERRFL=  | 000000 | \$MADR3    | 001236 | \$TKS    | 001144 | \$11      | 002440 |
| TYPE =  | 104401 | \$ERROR   | 016010 | \$MADR4    | 001242 | \$TN =   | 000023 | \$110     | 005320 |
| TYPOC = | 104402 | \$ERRPC   | 001116 | \$MAIL     | 001174 | \$TPB    | 001152 | \$111     | 005360 |
| TYPON = | 104404 | \$ERRTB   | 001254 | \$MAMS1    | 001224 | \$TPFLG  | 001157 | \$112     | 005364 |
| TYPOS - | 104403 | \$ERTTL   | 001112 | \$MAMS2    | 001230 | \$TPS    | 001150 | \$113     | 005452 |
| VECTOR  | 014160 | \$ESCAP   | 001162 | \$MAMS3    | 001234 | \$TRAP   | 017142 | \$114     | 005460 |
| WAIT    | 012620 | \$ETABL   | 001214 | \$MAMS4    | 001240 | \$TRAP2  | 017164 | \$115     | 005476 |
| WHICHB- | 000002 | \$ETEND   | 001254 | \$MBADR    | 001002 | \$TRP =  | 000012 | \$116     | 005514 |
| WRAP =  | 000020 | \$FATAL   | 001176 | \$MCALL=   | 000000 | \$TRPAD  | 017176 | \$117     | 005620 |
| XMITIE  | 000100 | \$FFLG    | 016006 | \$MFLG     | 016004 | \$TSK0 = | 000251 | \$12      | 002510 |
| XMITRD= | 000200 | \$FILLC   | 001156 | \$MNEW     | 015531 | \$TSK1 = | 000250 | \$120     | 005616 |
| XMITOS  | 007454 | \$FILLS   | 001155 | \$MSGAD    | 001210 | \$TSK2 = | 000254 | \$121     | 005562 |
| XMIT1S  | 007534 | \$F\$AND= | 000310 | \$MSGLG    | 001212 | \$TSK3 = | 000253 | \$122     | 005614 |
| XMIT2S  | 007614 | \$F\$BAD  | 000401 | \$MSGTY    | 001174 | \$TSK4 = | 000126 | \$123     | 005576 |
| XMIT3S  | 007674 | \$F\$BLA= | 000170 | \$MSWR     | 015520 | \$STSM   | 001004 | \$124     | 005614 |
| XOSRV   | 010734 | \$F\$CAS= | 000150 | \$MTYP1    | 001225 | \$STSTM  | 001102 | \$125     | 005612 |
| X1SRV   | 011054 | \$F\$DEC  | 000220 | \$MTYP2    | 001231 | \$TTYIN  | 015476 | \$126     | 005614 |
| X2SRV   | 011174 | \$F\$GOO= | 000400 | \$MTYP3    | 001235 | \$TYPDS  | 016470 | \$127     | 005620 |
| X3SRV   | 011314 | \$F\$IF   | 000110 | \$MTYP4    | 001241 | \$TYPE   | 014504 | \$13      | 002536 |

|       |        |       |        |       |        |       |        |      |        |
|-------|--------|-------|--------|-------|--------|-------|--------|------|--------|
| \$130 | 005652 | \$163 | 006646 | \$215 | 010524 | \$25  | 003300 | \$51 | 003754 |
| \$131 | 005712 | \$164 | 006646 | \$216 | 010534 | \$250 | 012640 | \$52 | 003750 |
| \$132 | 005734 | \$165 | 006774 | \$217 | 010554 | \$251 | 012670 | \$53 | 003752 |
| \$133 | 005766 | \$166 | 007022 | \$22  | 003016 | \$252 | 012656 | \$54 | 003756 |
| \$134 | 005760 | \$167 | 007054 | \$220 | 010614 | \$253 | 012654 | \$55 | 004024 |
| \$135 | 005766 | \$17  | 002700 | \$221 | 010614 | \$254 | 012666 | \$56 | 004062 |
| \$136 | 006030 | \$170 | 007112 | \$222 | 010632 | \$255 | 014360 | \$57 | 004102 |
| \$137 | 006036 | \$171 | 007202 | \$223 | 010644 | \$256 | 014360 | \$6  | 002334 |
| \$14  | 002564 | \$172 | 007202 | \$224 | 010654 | \$26  | 003126 | \$60 | 004124 |
| \$140 | 006052 | \$173 | 007244 | \$225 | 010726 | \$27  | 003166 | \$61 | 004172 |
| \$141 | 006052 | \$174 | 007244 | \$226 | 010732 | \$3   | 002130 | \$62 | 004206 |
| \$142 | 006134 | \$175 | 007422 | \$227 | 011046 | \$30  | 003222 | \$63 | 004220 |
| \$143 | 006152 | \$176 | 010024 | \$23  | 003046 | \$31  | 003222 | \$64 | 004404 |
| \$144 | 006166 | \$177 | 010044 | \$230 | 011052 | \$32  | 003260 | \$65 | 004432 |
| \$145 | 006240 | \$2   | 002066 | \$231 | 011166 | \$33  | 003256 | \$66 | 004550 |
| \$146 | 006224 | \$20  | 002726 | \$232 | 011172 | \$34  | 003266 | \$67 | 004554 |
| \$147 | 006222 | \$200 | 010132 | \$233 | 011306 | \$35  | 003346 | \$7  | 002362 |
| \$15  | 002614 | \$201 | 010170 | \$234 | 011312 | \$36  | 003350 | \$70 | 004724 |
| \$150 | 006240 | \$202 | 010174 | \$235 | 012614 | \$37  | 003416 | \$71 | 004752 |
| \$151 | 006240 | \$203 | 010232 | \$236 | 012616 | \$4   | 002154 | \$72 | 005032 |
| \$152 | 006266 | \$204 | 010232 | \$237 | 012466 | \$40  | 003440 | \$73 | 005104 |
| \$153 | 006300 | \$205 | 010274 | \$24  | 003052 | \$41  | 003476 | \$74 | 005106 |
| \$154 | 006312 | \$206 | 010316 | \$240 | 012570 | \$42  | 003500 | \$75 | 005116 |
| \$155 | 006324 | \$207 | 010326 | \$241 | 012504 | \$43  | 003546 | \$76 | 005114 |
| \$156 | 006374 | \$21  | 002756 | \$242 | 012512 | \$44  | 003566 | \$77 | 005322 |
| \$157 | 006464 | \$210 | 010360 | \$243 | 012530 | \$45  | 003624 | -    | 017222 |
| \$16  | 002652 | \$211 | 010374 | \$244 | 012602 | \$46  | 003626 | .\$X | 001000 |
| \$160 | 006476 | \$212 | 010450 | \$245 | 012676 | \$47  | 003674 |      |        |
| \$161 | 006750 | \$213 | 010554 | \$246 | 012676 | \$5   | 002314 |      |        |
| \$162 | 006574 | \$214 | 010510 | \$247 | 012644 | \$50  | 003712 |      |        |

. ABS. 017222 000 CON RO ABS LCL D

ERRORS DETECTED: 0

Z:CVDLAB/I,Z:CVDLAB.SEQ=SPMAC.SML,CVDLAB.P11  
 RUN-TIME: 101 93.7 SECONDS  
 RUN-TIME RATIO: 312/195 1.5  
 CORE USED: 33k (65 PAGES)